Trabajo Final de Grado

Grado en Ingeniería en Vehículos Aeroespaciales (GREVA)

PROYECTO DE DESARROLLO DE UN ENTORNO DE VISUALIZACIÓN Y MANDO (SCADA) DE UN BANCO DE ENSAYOS DE ADCS PARA CUBESATS UTILIZANDO PROCESSING

MEMORIA

29 de junio de 2020

Autor: Márquez González, David

Director: González Diez, David

Co-Director: Esquerra Llucià, Ignasi

Convocatoria: 30 de Junio de 2020

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa

Declaración de honores

I declare that,

the work in this Master Thesis / Degree Thesis (choose one) is completely my own work,

no part of this Master Thesis / Degree Thesis (choose one) is taken from other people's work without giving them credit,

all references have been clearly cited,

I understand that an infringement of this declaration leaves me subject to the foreseen disciplinary actions by The Universitat Politècnica de Catalunya - BarcelonaTECH.

David Márquez González		30/06/2020
Student Name	Signature	Date

Title of the Thesis : PROYECTO DE DESARROLLO DE UN ENTORNO DE VISUALIZACIÓN Y MANDO (SCADA) DE UN BANCO DE ENSAYOS DE ADCS PARA CUBESATS UTILIZANDO PROCESSING

Abstract

El objetivo principal de este proyecto es desarrollar un entorno que permita controlar y visualizar un cubesat. En concreto, este software se aplica a un banco de ensayos de ADCS, para que se puedan visualizar los datos más relevantes en pantalla, y se puedan enviar comandos al cubesat. También se enlaza el software con un propagador orbital, con el fin de mostrar la órbita que seguiría el modelo en pantalla.

Como no se dispone de un banco de pruebas de ADCS ni de un ADCS en formato físico, se han tenido que hacer unos ajustes al proyecto. Los datos de entrada no serán los medidos por el ADCS y por el cubesat. En su lugar los provendrán de un archivo en formato .csv que contendrá los valores que leerá el SCADA. Además, estos datos están ordenados según los estándares que se describen en la CCSDS 502.0-B-2. En lo referente al sistema de control del cubesat, este no envía ninguna señal ni valores nuevos al procesador del ADCS para orientar al cubesat con la actitud deseada. En su lugar, se han creado unas variables que simulan la actuación del hardware del ADCS. Estas variables actúan sobre los valores leídos del archivo de datos, cambiando así los valores leídos por pantalla.

El SCADA se ha diseñado en Processing y cuenta con cinco elementos. Estos se encargan de la visualización numérica de los datos del cubesat, la visualización 3D de la orientación del cubesat, el dibujo de la órbita en forma de traza terrestre, el control de la orientación y la posición del cubesat, y la zona de introducción de los valores de posición de las estaciones terrestres y de las zonas de estudio.

El proyecto se ha llevado a cabo de forma satisfactoria y se ha logrado diseñar un entorno que cumple las funciones requeridas. Además se ha solventado la ausencia del banco de pruebas y del ADCS, y la simulación de los datos es cercana a la real. Realizando unos pequeños cambios al código en cuanto a la obtención de datos y al envío de comandos, el proyecto creado se podría adaptar para ser integrado en un banco de pruebas de ADCS real.

Índice

Lista de figuras

Lista de Tablas

Lista de acrónimos

- ADCS: Attitude Determination and Control System
- SCADA: Supervisory Control And Data Acquisition
- CCSDS: Consultative Committee for Space Data Systems
- NASA: National Aeronautics and Space Administration
- MCT: Mission Control Software
- ISS: International Space Station
- LVLH: Local Vertical Local Horizontal

1. Introducción

En esta sección se introduce el proyecto, dividido en los objetivos que tiene que cumplir, el alcance del proyecto detallando qué se va a hacer y hasta donde se llegará, los requisitos del proyecto y la justificación explicando por qué es necesario este proyecto.

1.1. Objetivos del proyecto

En este proyecto se desarrollará, utilizando el lenguaje Processing, un entorno que permita visualizar y controlar un modelo de ingeniería de un cubesat. En concreto se aplicará a un banco de ensayos de ADCS (Attitude Determination Control System), de forma que el software desarrollado permita tanto visualizar datos relevantes del estado del cubesat (orientación, estado de la bateria, insolación, temperatura,etc) como el envío de comandos al mismo. También se contempla enlazar el sofware con un propagador orbital, de manera que se pueda mostrar la órbita que seguiría el modelo.

1.2. Alcance del proyecto

En este proyecto se desarrollará un entorno SCADA de visualización y control en Processing. Con el fin de conseguir este objetivo, los aspectos más importantes del proyecto se desarrollan a continuación:

- Estudio y búsqueda de información sobre los sistemas SCADA o interfaces gráficas en cubesats.
- Estudio y aprendizaje mediante tutoriales del lenguaje de programación Processing para obtener los conocimientos necesarios para desarrollar el proyecto.
- Este entorno permitirá visualizar la posición del cubesat, dibujando este como un cubo en la pantalla orientado según los datos de actitud obtenidos. A su vez, el color del cubesat cambiará cuando la batería de este se encuentre baja.
- También se mostrarán en pantalla los datos de: orientación, estado de la batería, insolación y temperatura. La orientación se enseñará en forma de tres ángulos, cada uno respecto de los ejes X, Y y Z. El estado de la batería se enseñará en forma de porcentaje de batería restante. La insolación se enseñará con el mensaje 'cubesat insolado' o 'cubesat no insolado'. La temperatura se mostrará en forma de valor en grados Celsius.
- También se mostrarán en pantalla los datos de órbita del cubesat, los cuales son. longitud, latitud, velocidad y altura. Se utilizarán las medidas estandarizadas para satélites las cuáles son º, km/s y km.
- Dado que no hay la posibilidad de contar con un cubesat, los datos del proyecto no vendrán de un banco de ensayos ADCS, sino que se simularán a partir de unos datos inven-

tados en un archivo Excel.

- El entorno permitirá controlar la orientación del cubesat a partir de unos botones que permitirán modificar los ángulos de este.
- El entorno también permitirá controlar la posición del cubesat a partir de unos botones que modificarán la longitud y la latitud de este.
- La trayectoria del satélite será visualizada en pantalla, en este caso se dibujará su traza terrestre, que consiste en el dibujo de todas las posiciones del satélite, entendiendo por posición la intersección de la línea que forman el inicio del eje de coordenadas terrestre y el satélite con la superficie terrestre.

Por otro lado, dado la complejidad del cálculo de la traza terrestre, los valores que se introducirán en el archivo de datos serán la latitud y longitud del cubesat como si ya se hubiesen calculado anteriormente a partir de los datos en formato TLE.

1.3. Requisitos

El principal requisito de este proyecto es programar un sistema SCADA para cubesats, el cual se consigue completando los objetivos marcados en la sección anterior 1.2.
Otro requisito es que el entorno del proyecto tiene que replicar con la mayor fiabilidad los datos que llegarían de un ADCS dado que no se cuenta con uno de estos. Para esto se mirarán los documentos del CCSDS sobre recomendaciones para los estándares del sistema de datos espaciales. Además los valores de latitud y longitud dela archivo de datos serán obtenidos a partir de datos descargados de satélites existentes.
Al Processing ser un programa gratis, no requiere de licencia y puede ser descargado desde cualquier computadora sin coste.

1.4. Justificación y utilidad

Los sistemas de determinación y control de actitud son necesarios en el cubesat dado que permiten estabilizar y orientar el cubesat hacia la dirección introducida. Por ejemplo se utiliza para orientar los paneles solares hacia el Sol para así obtener la mayor potencia posible, obteniendo así una mayor carga de batería. También se utilizan si el satélite tiene que apuntar hacia una determinada zona terrestre de la cual se está realizando un estudio. Esto provoca que los ADCS sean una parte crítica de los cubesats, dado que si fallan pueden provocar una desestabilización del cubesat, o una mala orientación de este. Después de simular y aprobar el diseño del ADCS, su funcionamiento se tiene que comprobar en formato físico. Para llevar a cabo estas pruebas hay que testearlos en un banco de pruebas y comprobar si su funcionamiento y calibración son correctas.

Para llevar estas pruebas a cabo es necesario ver los datos de orientación y posición obtenidos por el cubesat, dado que estos son los que el ADCS determina y controla. Esta visualización

es necesaria porque al ser un banco de pruebas, se necesita ver si las medidas obtenidas por el ADCS corresponden con las reales, además de como el cubesat es afectado por la gravedad zero o por el campo magnético terrestre. También es necesario un sistema de control del cubesat, para ir cambiando los datos que se envían al ADCS y así comprobar su correcto funcionamiento para diferentes situaciones, ya sea a corto plazo o a largo plazo. Para llevar a cabo estas tareas, se desarrolla este proyecto, dado que se diseña un entorno que sea capaz de mostrar los datos que se obtienen de un banco de pruebas ADCS de forma gráfica y analítica, y a la vez este entorno tiene que ser capaz controlar el cubesat de forma remota, enviando datos de posición y orientación al ADCS para que este oriente y pòsicione el cubesat con los datos enviados.

Específicamente el presente proyecto se aplicará al banco de ensayos de ADCS de la Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa (ESEIAAT). Este proyecto permite al cuerpo docente y alumnado testear los ADCS y los algoritmos creados más allá de su simulación en un pantalla, dado que se permite controlar el banco de pruebas, a la vez que se visualizan los resultados de los algoritmos. El banco de pruebas al cual se aplicarán es el desarrollado por otro proyectista en su tesis de final de grado titulada 'Estudio, Diseño y construcción de una plataforma pivotante para ensayo del sistema de control de actitud (ACS) de un cubesat'.

2. Desarrollo

En esta sección se presenta el estado del arte con una breve explicación de los ADCS y los bancos de pruebas. También se plantean las posibles soluciones, y se justifica la elección de la solución propuesta.

2.1. Estado del arte

Hoy en día el espacio es uno de los sectores donde más dinero se invierte en investigación y proyectos. Esto permite el desarrollo de nuevas tecnologías relacionadas con el espacio, generando así una innovación en este campo.

Esta innovación se puede ver en el diseño de satélites, donde anteriormente todos eran grandes y voluminosos. Ahora, aún se utilizan este tipo de satélites, pero también han aparecido los cubesats, satélites de medida reducida que son simples, compactos, y diseñados para tareas específicas. Estas características hacen que el coste de un cubesat se vea reducido notablemente en comparación con un satélite convencional, permitiendo así que centros educativos, grupos de investigación y pequeñas empresas puedan investigar y conocer el sector de los satélites y el espacio, dado que el coste de los cubesats se encuentra dentro de su capacidad monetaria.

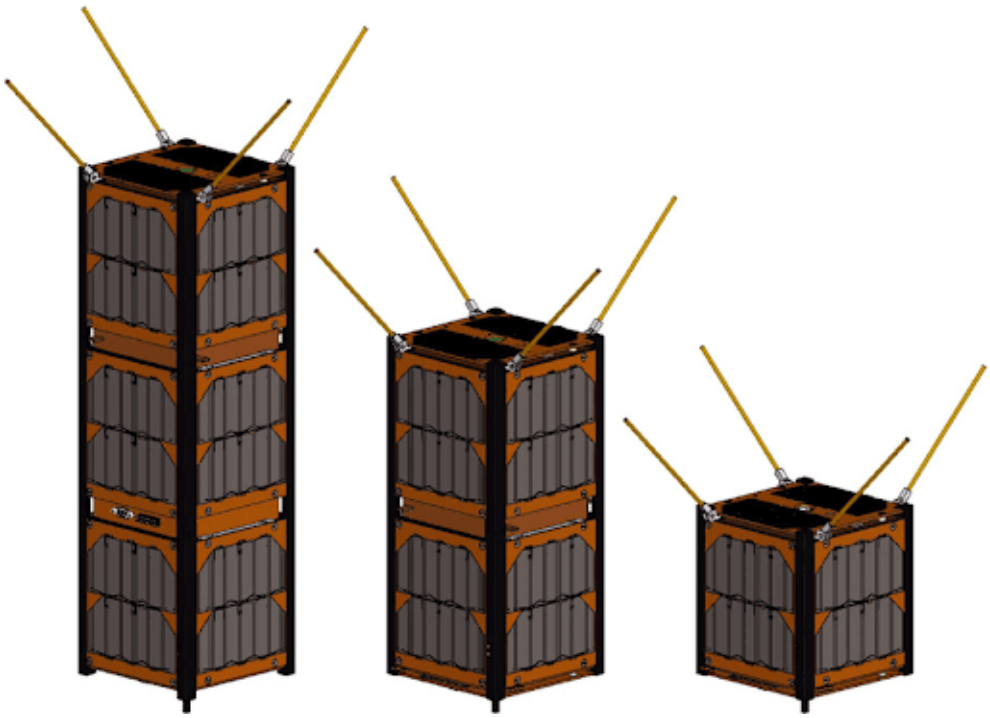

Figura 1: Cubesats de 3U, 2 U y 1U [1]

Una parte esencial de los cubesats es el sistema de determinación y control de actitud. Esto se debe a que el ADCS se encarga de aportar estabilidad y precisión de apuntado al cubesat, garantizando así la estabilidad de la carga útil y las antenas, haciendo que estos sistemas funcionen correctamente. También se encarga de que el cubesat tenga los paneles solares orientados para obtener la máxima radiación solar, y que los sensores y cámaras del cubesat estén orientados hacia la zona terrestre que se desea estudiar. Los elementos de los ADCS son varios, entre los cuales se encuentran el procesador, las ruedas de reacción, magnetórquers, magnetómetros y ruedas de reacción. Estos conforman el hardware del ADCS mostrado en 2. [3]

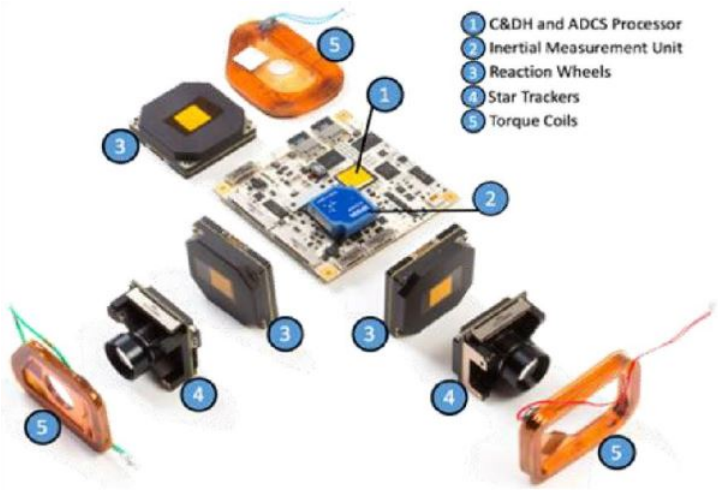

Figura 2: Hardware de un ADCS [2]

Otra parte importante de un ADCS es el software, que lleva integrado los algoritmos de estimación de actitud, que se encargan de transformar los datos que le llegan de los sensores en números. El software también controla el diseño del bucle de control, que se encarga de comparar los valores de actitud obtenidos por el algoritmo con los valores introducidos por el usuario, y envía órdenes de control al hardware en el caso de que estos no coincidan para conseguir que la actitud obtenida y la introducida sea la misma. Hay que incluir en las partes críticas de los ADCS las pruebas de estos algoritmos, tanto en simulaciones como físicamente.

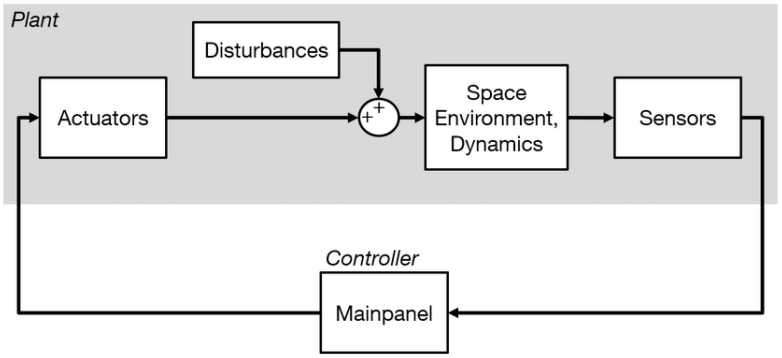

Figura 3: Loop de control de un ADCS [4]

Primero se lleva a cabo la simulación del modelo de ADCS en el ordenador, para ver si el diseño funciona, y para comprobar los algoritmos del cubesat. Luego se llevan a cabo pruebas físicas, que comprueban el funcionamiento del ADCS a corto y largo plazo, dado que la vida útil de un cubesat suele comprender entre 0,5 y 1 año. Además los tests físicos también comprueban la conexión del sistema de control con el hardware, y se asegura de que este actúa correctamente, asegurando que la actitud en la que se posiciona el cubesat es la deseada, y que el algoritmo de estimación funciona correctamente y por lo tanto la actitud calculada por este algoritmo corresponde con la real.

Las pruebas físicas de los sistemas ADCS se llevan a cabo en un banco de pruebas que intenta replicar las condiciones en las que se encontraría el cubesat cuando esté en órbita. Está formado por un conjunto de seis bobinas de cobre, colocadas dos en cada eje y paralelas entre sí, como se muestra en 4. Cambiando la corriente eléctrica que pasa por el interior de estas bobinas, se genera un campo magnético constante que replica el campo magnético terrestre.

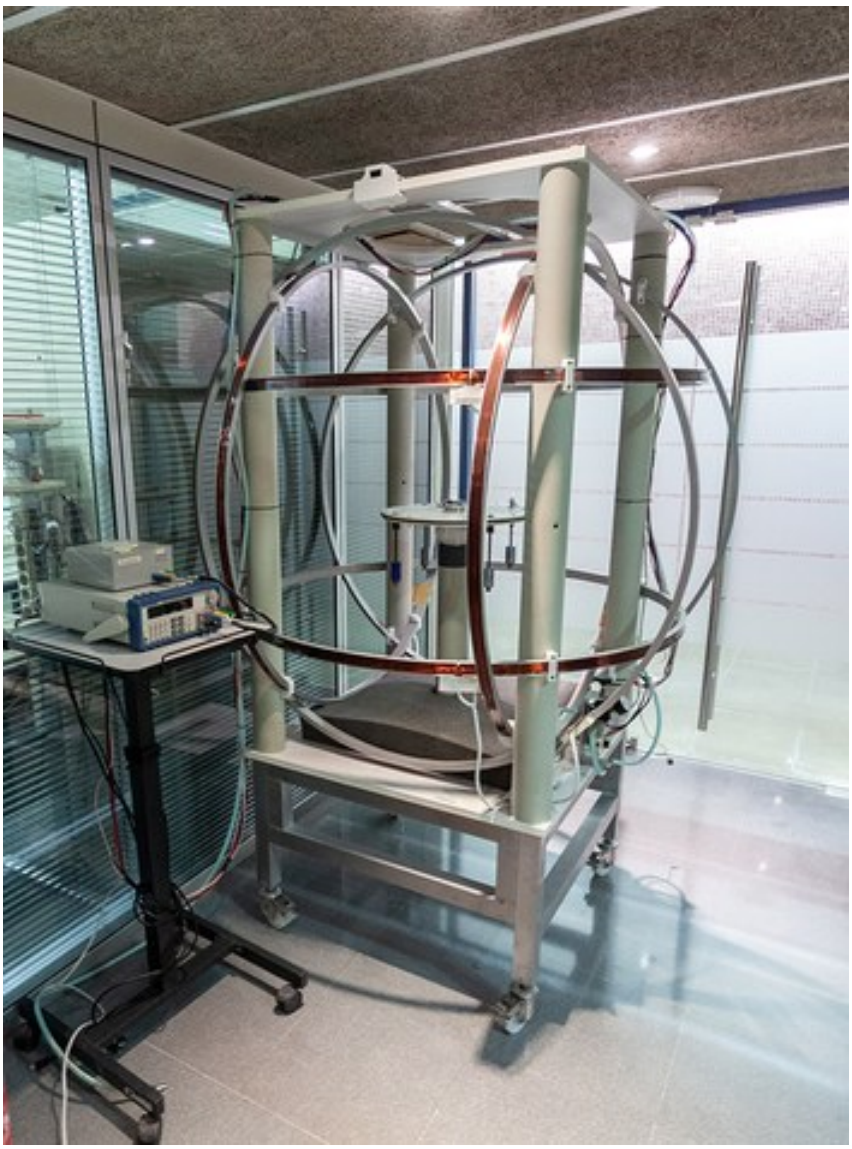

Figura 4: Banco de pruebas de ADCS del NanoSat Lab [5]

Además de las bobinas, los bancos de pruebas también incluyen una plataforma donde se depositan los sistemas o dispositivos que hay que probar, en este caso el ADCS. Esta superficie está montada sobre un cojinete de aire, conocido también como 'air bearing'. Estos son cojinetes de forma esférica, dado que es la forma más óptima porque permite el movimiento de rotación en torno a cualquiera de los tres ejes. Una vez se enciende el banco de pruebas, se expulsa aire en la parte inferior del cojinete, creando así un colchón de aire bajo la superficie de pruebas donde se encuentran los elementos a los que se le van a realizar los tests [6]. El colchón de aire permite que la plataforma se pueda mover libremente y sin fricción, intentando simular así las condiciones de gravedad cero. Al encontrarse el cojinete flotando sobre el colchón de aire, el momento angular creado por las ruedas de reacción y los magnetórquers cambia la orientación del cubesat hacia la deseada. La figura 5 muestra la estructura que tiene el banco de pruebas.

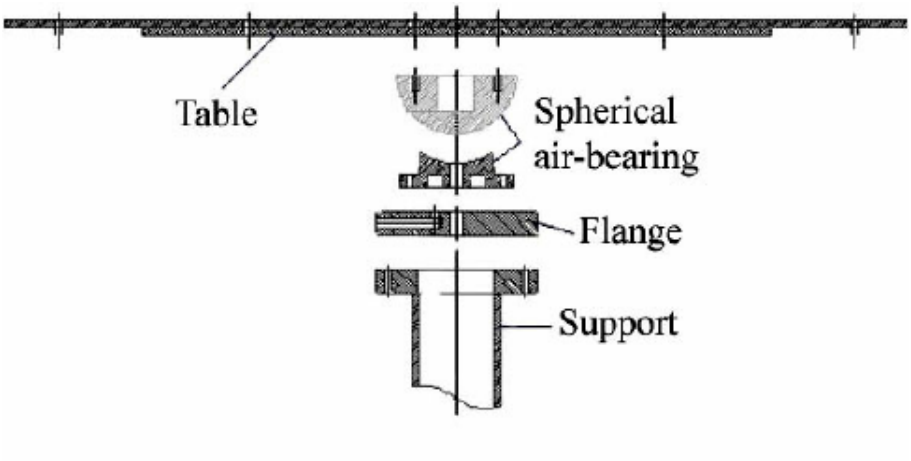

Figura 5: Elementos de la plataforma del banco de pruebas [7]

Los datos obtenidos a partir de los instrumentos de medida y de los algoritmos de estimación de actitud del ADCS necesitan ser enviados a otro sistema para que los almacene y los muestre en pantalla. Esto permite ver la evolución del ADCS a corto y largo plazo, ya que se ve el valor instantáneo de la actitud del cuebsat, pero también se puede observar la evolución de la actitud de este en el tiempo. También se necesita un sistema que controle la actitud del cubesat, es decir, que envíe valores de actitud al cubesat y que este mediante el hardware del ADCS se posicione en esa actitud introducida. Para llevar a cabo estas funciones se utiliza un SCADA, al cual se envían los datos que se han obtenido de actitud, posición, y otros valores generales del cubesat. El SCADA los procesa y los muestra, además que envía órdenes al ADCS según si se necesita cambiar la actitud del cubesat. Dado que un SCADA puede controlar más de un sistema, también muestra una alarma en el caso de que haya algún error o valor crítico en el cubesat, como podría ser la temperatura o la batería de este. La arquitectura de un SCADA explicada anteriormente se muestra en 6.

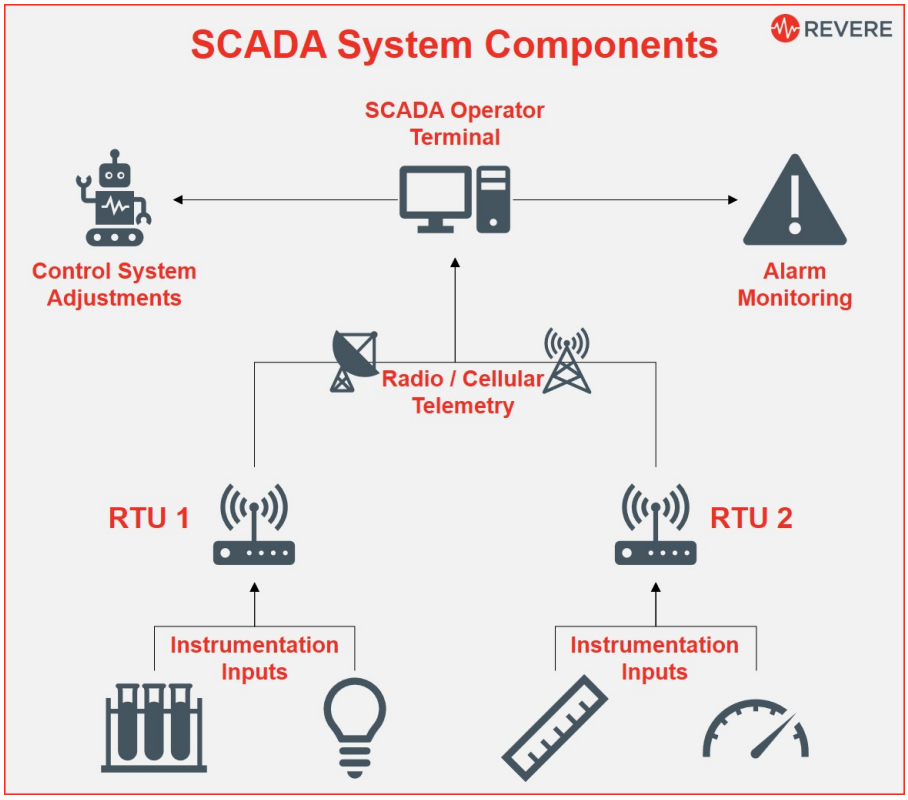

Figura 6: Elementos que forman un SCADA [8]

La arquitectura del SCADA es como la mostrada en 6 porque esta permite visualizar y controlar otro sistema o subsistema de manera automática. Los SCADA son utilizados sobre todo en el sector industrial, donde hay grandes cadenas de montaje y se requiere de un sistema para poder detectar posibles fallos en esta más fácilmente. Actualmente estos sistemas se están extendiendo por todos los sectores, hasta llegar al sector espacial. Muchos satélites convencionales cuentan con un sistema autónomo que permite controlar y mostrar las características del satélite en tiempo real. El uso de SCADAs también se ha extendido a los cubesats, donde cada equipo de investigación utiliza un diseño diferente para poder visualizar y controlar su cubesat de forma remota tanto en los periodos de testing como cuando ya se encuentra en órbita.

La realización del proyecto busca crear un sistema SCADA para utilizarlo en un banco de pruebas de ADCS. Esto permitiría visualizar las características del satélite de una forma gráfica y numérica a la vez, junto con la posibilidad del control de este en el caso que fuese necesario, ayudando así a la calibración delalgoritmo del ADCS, y existiendo también la posibilidad de utilizar este SCADA en misiones espaciales reales. El SCADA de la figura 8 es un ejemplo de un SCADA de un cubesat, donde se muestran la traza terrestre y la variación del enlace Estación terrestre-Cubesat debido al efecto Doppler del cubesat, junto con diferentes parámetros relevantes de este como serían los datos del cubesat y los datos orbitales.

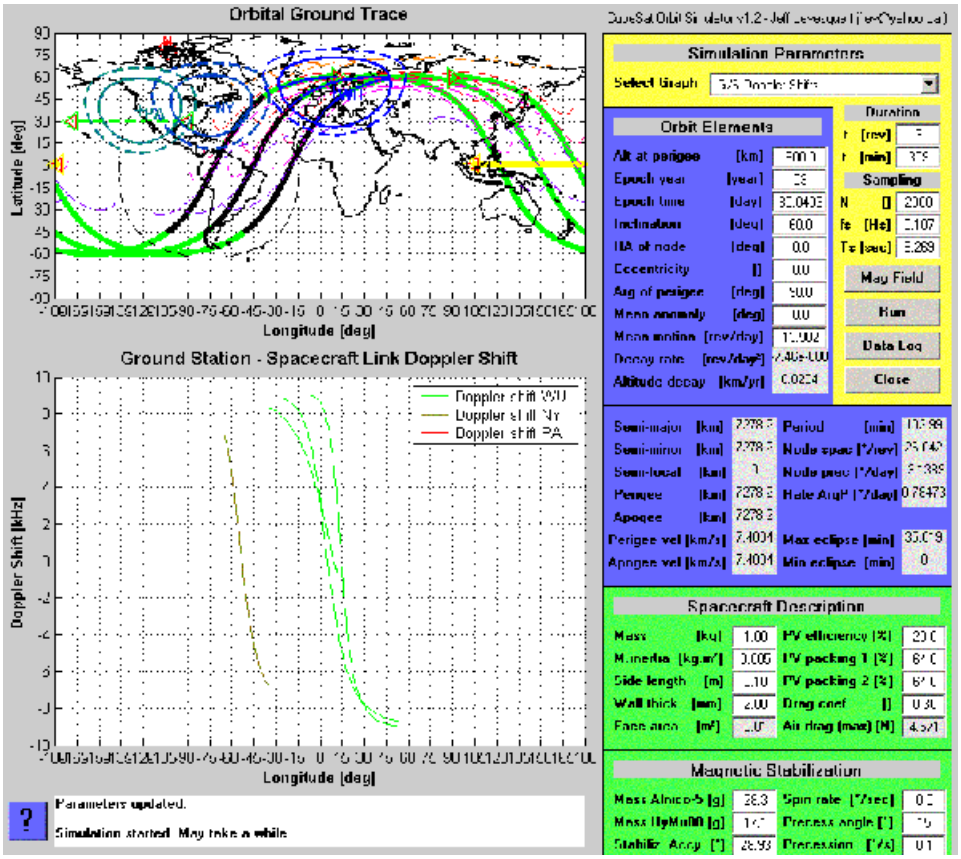

Figura 7: SCADA de un cubesat [9]

2.2. Planteamiento y selección de alternativas

Como se comenta anteriormente en 1.4, es necesario un sistema que muestre que la calibración de los ADCS es la correcta, y que los algoritmos de estos funcionan correctamente. También es necesario un sistema de control que envíe señales al ADCS sobre la actitud que este debe tomar.

Una posible alternativa sería el software de la NASA: Open MCT. Consiste en un marco de control de misión de última generación que se encarga de la visualización de los datos. Ha sido desarrollado en el NASA's Ames Research Center en colaboración con el Jet Propulsion Laboratory. Este software es utilizado actualmente por la NASA en tareas como el análisis de datos de misiones espaciales, y también en pruebas de sistemas de rover experimentales. Este programa al ser de código abierto puede utilizarse para la planificación, operación y análisis de cualquier sistema que produzca datos de telemetría. Las ventajas que aporta Open MCT es que soporta operaciones distribuidas, da acceso a datos en cualquier lugar, visualización de datos para análisis y reúne muchas de las funciones de las operaciones de la misión, haciendo así que todos los datos necesarios se muestren en una sola aplicación y no sea necesario que el operador vaya cambiando de aplicaciones constantemente para visualizar todos los datos. [10]

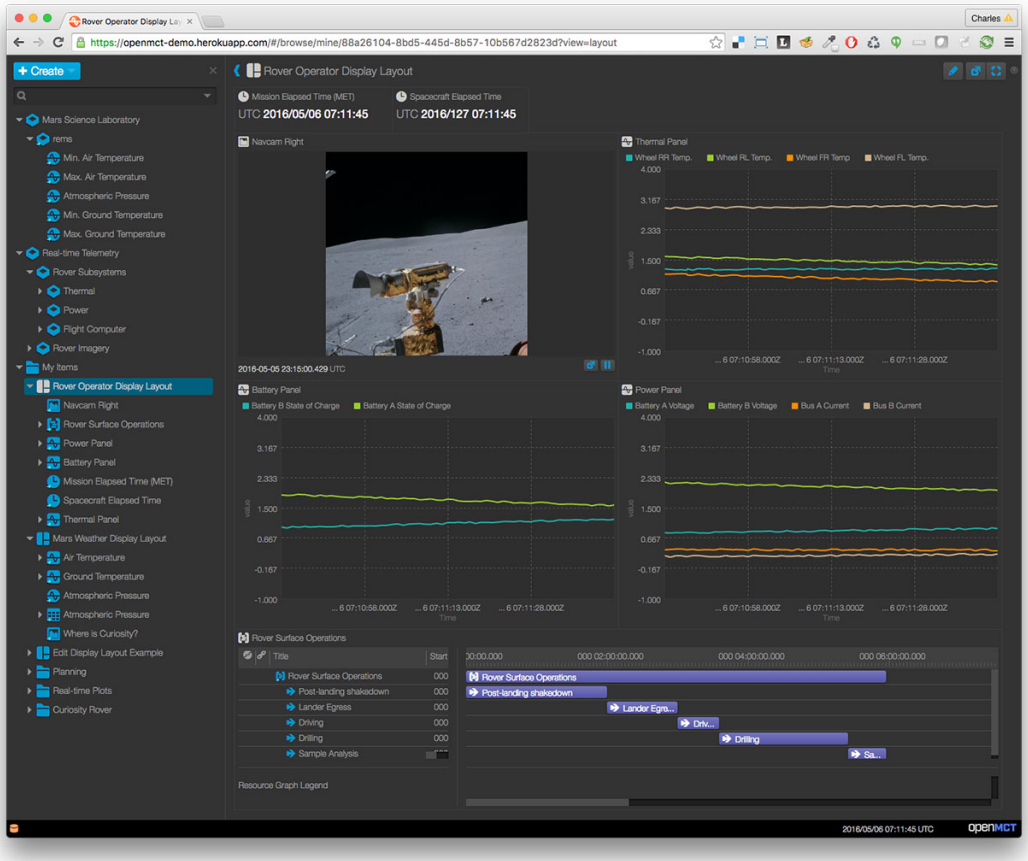

Figura 8: Software Open MCT de la NASA [10]

Como complemento a este sistema de visualización de datos, se necesitaría un sistema de control del ADCS, que enviase la actitud requerida por el usuario al ADCS, para que este coloque al cubesat en la orientación introducida. Hay varias soluciones a este problema, por ejemplo la creación de una mesa con controles y pulsadores, donde cada pulsador tenga una función, por ejemplo uno se encargaría de aumentar el ángulo X y otro de disminuirlo. Al apretarse el pulsador, se enviaría una señal al ADCS para que este cambie la actitud a la introducida en la mesa de control. Otra posible solución sería la creación de un software de control que haga la misma función que la anterior solución, pero en formato digital en lugar de físico. Este software podría mostrar en pantalla botones, que al hacer click sobre ellos se envíe una señal al ADCS sobre la actitud que se requiere. Este software, a diferencia del Open MCT, se encarga del control del satélite, por lo que se necesitarían tener ambos abiertos a la hora de operar el banco de pruebas.

Por último, una solución que fusiona ambas interfaces de visualización de datos y de control sería un entorno SCADA. El entorno SCADA permite visualizar los datos de forma analítica a la vez que gráfica, obteniendo los datos del ADCS y otros sensores del cubesat y presentándolos en pantalla. Esto es importante en los bancos de pruebas, dado que comprueba si las medidas obtenidas del sistema ADCS están bien, a la vez que permite visualizar otros datos críticos del satélite, por ejemplo la batería de este o su temperatura. La visualización gráfica es un añadido importante al banco de pruebas, dado que permite ver de forma más gráfica los datos de ac-

titud y de posición. Esto, al igual que el Open MCT, permite al usuario encontrar fallos en los ADCS de forma más rápida, a la vez que se da al usuario una representación más realista de la situación en la que se encuentra el cubesat, y no solo en forma numérica. Por otro lado, el SCADA permite controlar el satélite de forma remota. Esto es necesario, dado que una vez se han obtenido los datos del ADCS, se tienen que llevar a cabo las maniobras necesarias hasta que la orientación y posición del cubesat son las deseadas.

Comparando las alternativas propuestas, el hecho de tener la mesa de control en formato físico aporta un mayor coste al proyecto, dado que hay que comprar los componentes físicos. El formato físico también aporta problemas de espacio al banco de pruebas, ya que se requiere de un sistema de visualización de datos también, por lo que se necesitaría la mesa de control y un ordenador. Además al tener dos sistemas separados se necesitarían o dos operarios, o un operario que vaya cambiando de puesto. Otro factor en contra es que el hecho de tener el control y la visualización de datos en un mismo programa permite que al recibir datos del cubesat, estos se puedan cambiar utilizando los botones del SCADA, por lo que la acción se llevará a cabo más rápido dado que el control está integrado con la visualización y no se necesita cambiar de programa para cada vez que se quiera realizar una tarea diferente. Además al estar todo fusionado en un solo programa, el coste computacional será menor que utilizando dos programas, aumentando su eficiencia. Esto hace que el SCADA sea una mejor solución frente al Open MCT+software de control, ya que el SCADA permite que se corrijan antes los errores de posición y orientación, o que se cambie la posición y orientación del cubesat deseado de manera más rápida, al mismo tiempo que se continúan visualizando los datos que le llegan.

A partir de los argumentos descritos en el párrafo anterior, se elabora la siguiente tabla que presenta el Ordered Averaging Method. La G corresponde al peso que se le da a cada criterio, sobre 5. La P la puntuación de cada alternativa respecto a cada criterio, sobre 5 también. Los criterios escogidos son el coste del proyecto, la eficiencia, tanto de espacio como de operarios y memoria que utiliza, y la velocidad a la que se puede enviar órdenes de control al SCADA mientras se visualizan los datos.

Criterio	Peso	Open MCT + Mesa de control		Open MCT + Software de control		Entorno SCADA	
	G	P	PxG	P	PxG	P	PxG
Coste	2	3	6	5	10	5	10
Eficiencia	4	2	8	4	16	5	20
Velocidad	5	3	15	3	15	4	20
Suma PxG	11	29		41		50	
OWA		0,527		0,745		0,909	

Tabla 1: OWA de las posibles soluciones

Como se observa en 1, la solución más óptima es el desarrollo de un entorno SCADA para el banco de pruebas de ADCS, dado a que su coste es casi nulo, su eficiencia es elevada dado que solo requiere de un ordenador para funcionar, y su velocidad es alta ya que integra el control y la visualización en un mismo entorno. Una vez implementado el SCADA al banco de pruebas de ADCS, asegurando así que el ADCS funciona correctamente, se podría utilizar el sistema SCADA con los datos obtenidos del cubesat durante una misión espacial real. Esto permitiría visualizar y controlar su posición y orientación junto con otros datos, permitiendo por ejemplo cambiar la zona de estudio del cubesat. Además, dado que los cubesats tienen una vida útil corta, el SCADA permite llevar a cabo las maniobras más rápidamente, optimizando así su uso y obteniendo el máximo número de datos posibles de la misión.

3. Desarrollo del proyecto

En esta sección se desarrolla la solución propuesta anteriormente en 2.2. Se detallan los cambios que se han llevado a cabo para adaptar el proyecto debido a la ausencia de un banco de pruebas y un ADCS con los que trabajar. También se describe el interfaz SCADA del proyecto y el funcionamiento de cada elemento de este.

3.1. Arquitectura del SCADA

Como se describe en 1.1 el objetivo de este proyecto es diseñar una interfaz SCADA para un banco de pruebas ADCS. Esta interfaz depende de datos obtenidos del ADCS y de sensores del cubesat, los cuales son procesados y impresos por pantalla. Además el SCADA cuenta con un subsistema de control que envía los nuevos valores al procesador del ADCS para que envíe órdenes a los elementos de control del ADCS.

3.1.1. Arquitectura real

La arquitectura creada para este sistema SCADA se muestra en 9, desde la adquisición de datos de los sensores hasta el envío de órdenes y la visualización de estos datos. Esta arquitectura está creada a partir de 3 y 6.

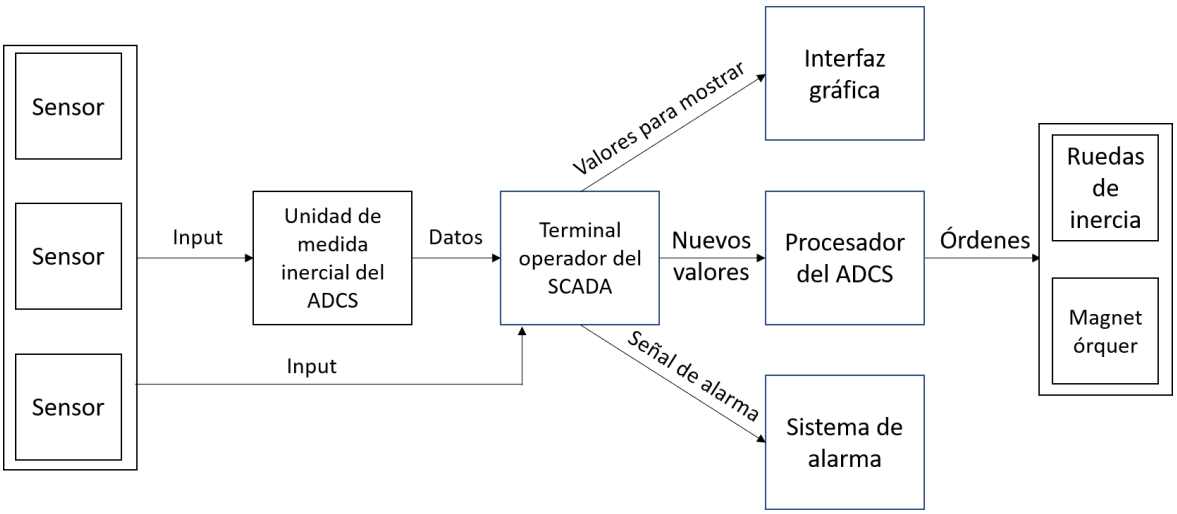

Figura 9: Arquitectura del sistema SCADA de un banco de pruebas de ADCS

La imagen anterior se puede dividir en tres fases, la captación de datos, el procesado en el terminal SCADA, y las órdenes de salida de este.

En lo que se refiere a la captación de datos, estos se harían a través de sensores que se encuentran en le ADCS. Los sensores del ADCS enviarían los inputs obtenidos a la unidad de medida

inercial del ADCS, el cual procesaría los datos obtenidos por el sensor y los transformaría en la actitud de este. Estos datos serían enviados al terminal operador del SCADA en forma de señal serial, como por ejemplo un archivo por vía bluetooth o wifi. A parte de los datos que le llegan del ADCS, al terminal también llegan otros datos obtenidos mediante otros sensores del cubesat, de sistemas secundarios diferentes al ADCS, como por ejemplo la batería o la temperatura de este. Al final los datos de entrada del SCADA corresponden a una mezcla entre los datos procesados por el ADCS y los datos obtenidos por otros sensores del cubesat.

Una vez los datos llegan por una vía serial al terminal SCADA, los datos que necesitan ser procesados se procesan con el fin de obtener datos en un formato que pueda ser mostrado por el SCADA. Un ejemplo sería la insolación del cubesat, donde los sensores envían una señal de voltaje de 1 o 0 y estos se tienen que transformar antes de ser graficados. Al ser un SCADA programado en Processing, hay que asegurarse de que el formato de entrada de todos los valores numéricos sea Float, y dado que la mayoría de sensores envían valores en formato String, estos valores necesitan ser transformados para que el sistema pueda mostrarlos correctamente.

Además de procesar los datos, el terminal también se encarga de recoger los cambios que se desean implementar en la actitud del cubesat, para ser enviados al ADCS. Para esto es necesario que el terminal SCADA incluya unos elementos de control en la interfaz gráfica, por ejemplo un botón o un bash, que una vez activados se encarguen de enviar las órdenes al procesador del ADCS para que este los transforme en acciones de los sistemas de control del ADCS, los cuales son las ruedas inerciales y los magnetórquers. Estos botones envían una señal parecida a la señal de datos que recibe el sistema SCADA, y esta también se envía por una vía serial al procesador del ADCS.

Una vez el terminal operador del SCADA ha procesado todos los datos, envía órdenes de control a diferentes sistemas para que estos actúen según los resultados obtenidos. Estos sistemas son:

- La interfaz gráfica, a la cual le llegan los valores que se deben mostrar procesados por el terminal operador del SCADA. Esta interfaz se encarga de mostrar los valores de forma numérica en pantalla, y también de graficar los valores de una forma más visual. Se va actualizando constantemente a la velocidad de llegada de los datos al SCADA. Esta interfaz además incluye botones y pulsadores para controlar la actitud que se desea obtener.

- El procesador del ADCS al cual le llegan los nuevos valores que se han introducido utilizando los elementos de control del terminal operador del SCADA. Este procesador compara los nuevos valores con los actuales, y crea una serie de órdenes que se necesitan llevar a cabo para cambiar la actitud del cubesat a la deseada. Estas órdenes ya procesadas son enviadas a las ruedas de inercia y magnetórquers que posicionarán el cubesat con la nueva actitud introducida.

- El sistema de alarma se encarga de avisar en el caso de que haya algún fallo crítica o de que esta se pueda producir en un futuro. Un ejemplo de esta podría ser la batería baja del cubesat, dado que al mostrar la alarma, el operario ve al momento que esta está cercana a agotarse y que por tanto las pruebas deberán finalizar en un corto periodo de tiempo, o que el cubesat necesita reorientarse para que las placas solares apunten hacia el Sol. Otro ejemplo podría ser la temperatura, dado que algún elemento puede sobrecalentarse y que-

marse, sobretodo en la unidad de medida inercial y el procesador del ADCS. Para evitar daños en este tipo de elementos tan sensibles, si la temperatura aumenta mucho también se podría enseñar una señal de alarma. Además, esto también indica que las pruebas del algoritmo utilizado sobrecargan demasiado el procesador, y que debe ajustarse, dado que la misión espacial espacial será a de larga durada, y el ADCS no aguantará en funcionamiento todo este tiempo.

3.1.2. Arquitectura adaptada

El diseño anterior mostrado en 9 contemplaba la disponibilidad de un banco de pruebas y de un ADCS para el proyecto. En este caso no ha sido posible disponer de dichos elementos, por lo cual es necesario aplicar ciertas adaptaciones a la arquitectura del SCADA. Los datos adquiridos en lugar de venir de sensores, se leen de un archivo .csv, que consiste en un archivo de datos separados por comas. Además los elementos de control del cubesat no envían los nuevos valores al procesador del ADCS, sino que actúan sobre ciertas variables de control para que el resultado sea la adición de estas variables al archivo de datos. La arquitectura del SCADA adaptada a las nuevas condiciones se muestra en 10.

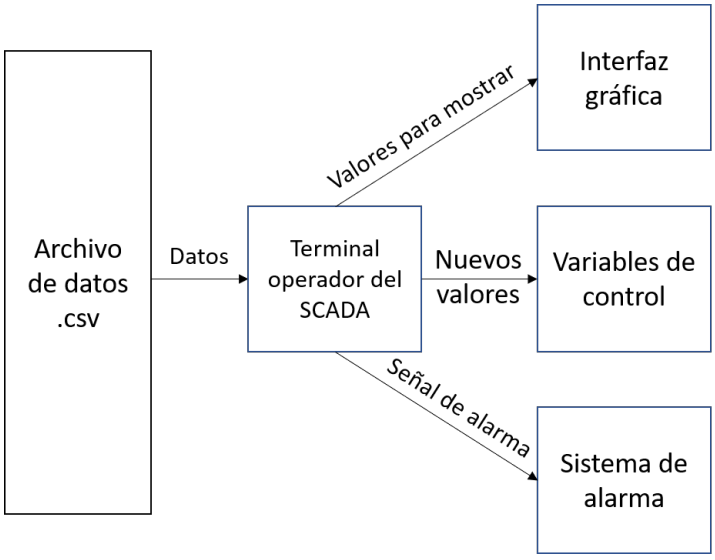

Figura 10: Arquitectura del sistema SCADA adaptado a las nuevas condiciones

Igual que en 3.1.1 la imagen anterior puede dividirse en tres fase, que serían el archivo de datos, y las órdenes de salida.

Haciendo referencia al archivo de datos, este es un archivo .csv que intenta simular los datos que le llegan al terminal operador del SCADA. Los datos se encuentran separados por comas, donde cada elemento sería un dato diferente que le llegaría, tanto ya procesado por la unidad de medida inercial de ADCS, como proveniente directamente del sensor del cubesat. En la primera fila se observa un listado de nombres. Este listado contiene la información de a qué corresponde cada dato en esa posición, para que el programa pueda leerlo más fácilmente, y para que aunque se cambie el orden de estos, el programa continúe funcionando. Cada fila es un array de datos,

es decir, que los datos de una misma fila corresponden todos a la misma ubicación temporal y se envían a la vez. Cada fila son archivos de datos nuevos que recibe el SCADA pasado cierto periodo de tiempo, es decir, que se simula el desarrollo temporal del ADC y del cubesat. Este se detalla en la siguiente sección 3.2.

El terminal operador del SCADA actúa es muy parecido al de 3.1.1 aunque no es completamente idéntico. La primera diferencia es que los datos los recoge del archivo .csv en vez de por una vía serial. Esto hace que en lugar de ir recibiendo los datos uno a uno, los reciba todos juntos y los vaya cargando línea a línea. Estos datos son procesados igual que se haría en el diseño anterior, dado que al provenir de un archivo .csv los datos siguen proveniendo en formato String. Por último un cambio que se hace al programa es la salida de los elementos de control. Anteriormente cuando se pulsaba un botón, este enviaba los datos al procesador del ADCS. Debido a que no se dispone de uno, se han creado variables para simular la actuación del hardware del ADCS. Esto hace que los nuevos valores no se envíen vía serial al procesador del ADCS, sino que al hacer click sobre los botones de control, se aumentan o disminuyen las variables de control, simulando la actuación del hardware ADCS y otorgando continuidad a la visualización de datos.

Al ser procesados los datos por el terminal operador del SCADA, se envían órdenes de control a diferentes sistemas para que estos actúen según los resultados obtenidos. Estos sistemas son:

- La interfaz gráfica de esta arquitectura, la cual es igual a la descrita en 3.1.1, y muestra los valores númerica y gráficamente. Los valores mostrados en pantalla ya se les ha aplicado la acción de control, lo que implica que cuando se muestra un valor en pantalla, este corresponde al valor del archivo de datos más el valor de la variable de control. Además los botones y pulsadores de control que se muestran son iguales a los que se mostrarían en la arquitectura anterior.

- Las variables de visualización , que se controlan mediante los botones de la interfaz gráfica, variando su valor cuando los botones se activan. Estas variables se mantienen permanentes desde el punto en que se activan, haciendo que haya una continuidad en la visualización de datos. Por ejemplo si se quiere aumentar el ángulo X en la fila 25, la variable sumará uno, y se aplicará a todos los datos a partir de la fila 25, y si se quiere disminuir 1 en la línea 45, los datos a partir de la línea 45 se mantendrán igual dado que antes se ha aumentado 1 y ahora se ha disminuido 1. Esto se encuentra explicado con más detalle en la sección 3.7.

- Por último el sistema de alarma es igual al sistema de 3.1.1, y se encarga de dar una señal de error en el caso de que alguno de los números introducidos en el archivo de datos corresponda a un valor crítico.

Una vez explicada la arquitectura del nuevo SCADA y las diferencias con el diseño real, se procede a detallar el diseño y funcionamiento del SCADA diseñado.

3.2. Archivo de datos

En esta sección se detallan los datos que contiene el archivo de datos, junto con sus unidades. Estos datos corresponden a la simulación de datos que llegarían del ADCS ya procesados y de otros sensores del cubesat. Los datos de este archivo son los elegidos dado que son los más cercanos a parecerse a los datos recibidos provenientes del ADCS. En la figura 11 se puede observar las cinco primeras líneas del archivo de datos, junto con la línea de encabezados. El proyecto cuenta con un total de 360 líneas con datos simulados.

```
epoch,longitud,latitud,altura,velocidad,anguloX,anguloY,anguloZ,masa,insolacion,temperatura,bateria
2020-06-05T00:12:40,-133.8275699,36.79835938,356044.0326,27743,0,0,180,1.33,1,0,100
2020-06-05T00:45:41,-132.9360836,36.1578537,355816.7062,27743,1,1,180,1.33,1,0,100
2020-06-05T01:20:42,-132.0600412,35.50979581,355589.7537,27743,2,2,180,1.33,1,0,100
2020-06-05T01:55:43,-131.1990053,34.85447643,355363.4538,27743,3,3,180,1.33,1,0,100
2020-06-05T01:20:43,-130.3525395,34.19217669,355138.0856,27743,4,4,180,1.33,1,0,100
```

Figura 11: Primeras líneas de larchivo de datos

Cada elemento de cada línea, separados por comas, corresponde a un dato diferente que le llega al SCADA. Estos datos son procesados por el SCADA y mostrados en pantalla numéricamente, a la vez que algunos son graficados. En el siguiente listado se encuentra detallado a qué corresponde cada dato, junto con su obtención:

- Epoch se refiere a la fecha de adquisición de datos. En este caso sigue el formato standard de la CCSDS 505.02-B-2, YYYY-MM-DDThh:mm:ss[.d→d], donde "YYYYçorresponde al año, "MM.ªl mes, "DD.ªl día, "T.es una constante que se utiliza para separar fecha de hora, "hh"son las horas, "mm"los minutos, y "ss"los segundos. El factor entre paréntesis ".d→dçorresponde a las décimas de segundo, pero este es opcional.[**?**]

- Longitud y latitud corresponden a los valores de posición X e Y sobre el mapa terrestre. Las unidades de estos elementos son grados magnéticos, y son positivos y negativos dependiendo de la posición del cubesat. Los valores positivos de latitud corresponden a posiciones al norte del Ecuador, y los valores negativos corresponden a posiciones al sur. La longitud es positiva cuando el cubesat se encuentra al este del Meridiano de Greenwich, y negativos cuando se encuentra al oeste. Esto se muestra más gráficamente en 12. Normalmente los valores son dados en formato TLE, a partir de los cuales se propaga la órbita y se calculan la longitud y latitud de esta y sus futuros valores. En este proyecto se ha optado por coger los datos ya procesados de otra misión espacial, concretamente los datos de longitud y latitud de este proyecto corresponden a los de la ISS. Esto simula que los datos en formato TLE adquiridos por el ADCS se procesan en un propagador orbital y que se transmiten al SCADA los datos ya procesados en formato de longitud y latitud.

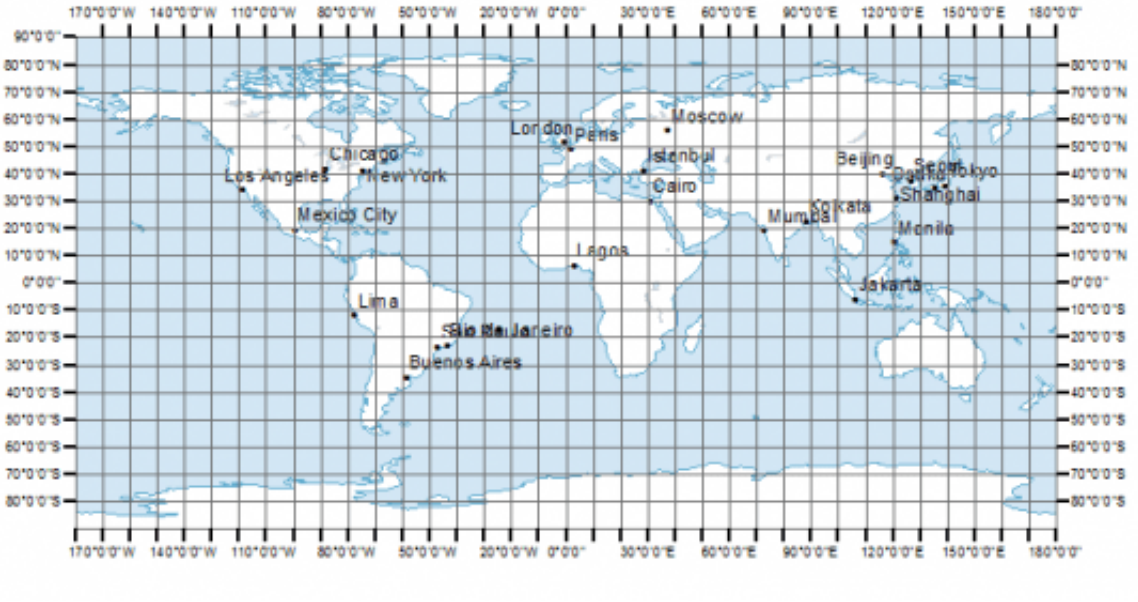

Figura 12: Signo y valor de la longitud y la latitud dependiendo de la posicón del cubesat [13]

- Los datos de altura corresponden a la distancia entre el satélite y la superficie terrestre, concretamente el nivel del mar. Se miden en kilómetros. En estos datos la mayoría son el mismo valor, aunque hay pequeñas variaciones para ver si el SCADA las lee bien, dado que la altura durante las pruebas del ADCS se mantendrá constante y durante la órbita también, con pequeñas variaciones en alguna maniobra.

- El elemento velocidad contiene los datos de la velocidad en el eje X del sistema de referencia, y su unidad son kilómetros por hora. Se muestra solo la velocidad en el eje X de la órbita dado que no hay variaciones en la velocidad en el eje Z, y en el eje Y solo habría variación en el momento de activar el control de la latitud o la longitud, dado que después se crea una nueva órbita y la velocidad vuelve a ser en el eje X del sistema de la nueva órbita. Por estas razones la velocidad mostrada es la del eje X solamente. Esta velocidad está cogida de la ISS, dado que también se han cogido los datos de longitud y latitud de esta.

- Los datos que ocupan las posiciones de anguloX, anguloY y anguloZ corresponden a los datos de los ángulos respecto a los ejes de la órbita. Los ejes de la órbita son los que estipulan los estándares de la CCSDS 500.0-G-3, en concreto el sistema de ejes LVLH. El sistema de ejes LVLH se muestra en 13, donde el eje X apunta en la dirección de desplazamiento del cubesat, el eje Z apunta hacia el centro de la tierra, y el eje Y completa el sistema para que el producto vectorial de la dirección X por la dirección Y de como resultado la dirección Z [12]. Estos datos simulan ser obtenidos por el procesador del ADCS, dado que los sensores le envían unos inputs al procesador y este los transforma en los ángulos. Los datos muestran un ángulo constante de 180º respecto al eje Z, y un ángulo inicial de 0º en los ejes X e y, que van aumentando de un grado en un grado hasta llegar a 45º, y despues se mantiene con un valor constante de 45º hasta el final del archivo de datos. Esto permite testear el algoritmo de graficar del SCADA en el caso de que los ángulos varíen como en

los ejes X e Y, o se mantengan constantes como en le eje Z.

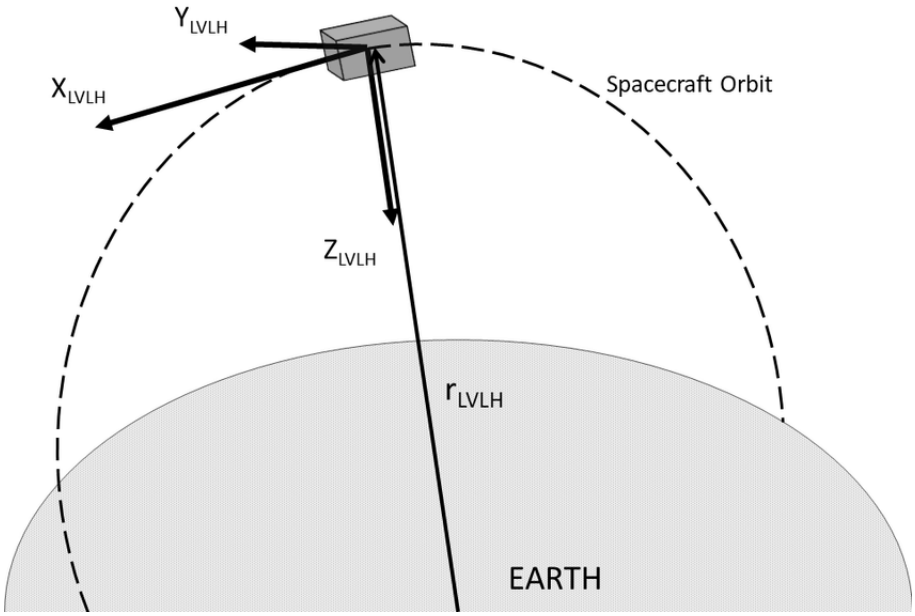

Figura 13: Sistema de ejes orbitales LVLH [14]

- El término masa muestra la masa total del cubesat y sus unidades son kilogramos. En el archivo se mantiene constante, dado que no se realizan variaciones en la masa durante las pruebas en el banco, pero se incluye dado que se podría equipar al cubesat con thrusters o otros elementos que requieran combustible.

- En la posición con el encabezado insolación se encuentran los datos que informan de si alguna cara del cubesat se encuentra recibiendo la energía solar o no. Esto se consigue mediante un sensor que envía una señal de 1 si se encuentra insolado, o 0 si no. Después el SCADA procesa estos datos y muestra por pantalla si el cubesat se encuentra insolado o no. En este caso en el archivo de datos varía de 1 a 0 dado que se prueba si el algoritmo y el procesado de los datos de insolación funcionan correctamente cuando el cubesat cambia de estar insolado a no estarlo.

- En penúltima posición se encuentran los datos de temperatura. Estos indican la temperatura del ADCS, por si algún subsistema se sobrecalienta, dado que esto podría provocar un fallo crítico en el ADCS y provocar un fallo en las misión. Los datos del archivo se encuentran en grados Celsius, y se mantienen constantes a 0ºC dado que esta es una temperatura aproximada, basada en los datos de otras misiones de cubesats.

- En último lugar del array de datos se encuentra la batería del cubesat. Se muestra en porcentaje restante, y es un valor crítico, dado que según la batería restante, la actitud es importante para controlar que los paneles solares reciban la mayor energía solar. Además, si este valor se encuentra por debajo del 5 % se envía una orden al sistema de alarma para mostrar en pantalla que queda poca batería. Los valores de la batería del archivo de da-

tos varían de 100 % al inicio, a 5 % después, para comprobar que el algoritmo de alarma funcione correctamente.

Los datos de este archivo están ordenados según los stándares de la CCSDS 502.02-B-2, donde se proporciona una visión general de los seis bloques lógicos en la sección de datos de OPM (Vector de estado, elementos keplerianos de oscilación, parámetros del satélite, posición / velocidad matriz de covarianza, parámetros de maniobra y parámetros definidos por el usuario). Además para cada elemento de datos se especifica:

- La palabra clave que se utilizará
- Una breve descripción del artículo
- Las unidades a utilizar
- Si el artículo es obligatorio u opcional

En la tabla siguiente 2 se pueden ver las especificaciones para los elementos de vector de estado, elementos keplerianos de oscilación, y parámetros del satélite[11]. Dado que no se contemplan matrices ni maniobras ya que el ADCS controla la actitud, los requisitos para estos no se tendrán en cuenta por lo tanto no se muestran.

Keyword	Description	Units	Obligatory
State vector components in the specified coordinate system			
COMMENT	(See 6.7 for formatting rules)	n/a	No
EPOCH	Epoch of the state vector and optional Keplerianelements. (See 6.5.9 for formatting rules.)	n/a	Yes
X	Position vector X-component	km	Yes
Y	Position vector Y-component	km	Yes
Z	Position vector Z-component	km	Yes
X_DOT	Velocity vector X-component	km	Yes
Y_DOT	Velocity vector Y-component	km	Yes
Z_DOT	Velocity vector Z-component	km	Yes
Osculating Keplerian Elements in the Specified Reference Frame (none or all parameters of this block must be given.)			
COMMENT	(See 6.7 for formatting rules)	n/a	No
SEMI_MAJOR_AXIS	Semi major axis	km	No
Eccentricity	Eccentricity	n/a	No
Inclination	Inclination	deg	No
RA_OF_ASCEND_NODE	Right ascencion of ascending node	deg	No
ARG_OF_PERICENTER	Argument of pericenter	deg	No
TRUE_ANOMALY	True anomaly	deg	No
GM	Gravitational coefficient (gravitational constant x central mass)	km^3/s^2	No
Spacecraft parameters			
COMMENT	(See 6.7 for formatting rules)	n/a	No
Mass	Spacecraft mass	kg	No
SOLAR_RAD_AREA	Solar radiation pressure area (Ag)	m^2	No
SOLAR_RAD_COEFF	Solar radiation pressure coefficient (Cr)	n/a	No
DRAG_AREA	Drag area (Ad)	m^2	No
DRAG_COEFF	Drag coefficient (Cd)	n/a	No

Tabla 2: Orden, descripción, unidades y obligatoriedad de los datos en formato OMM [11]

En el archivo de datos, los datos se encuentran siguiendo este orden. Los elementos kepleriano de oscilación se encuentran sustituidos por la actitud del satélite, y los datos de insolación sustituyen a los de radiación solar, dado que son parámetros del cubesat.

3.3. Diseño del SCADA

El diseño del SCADA se ha basado en la premisa de que se debe mostrar los datos numéricamente, además de mostrar la actitud y posición del satélite gráficamente, y que deben haber elementos de control de estas. En concreto se requería un apartado donde se mostrasen los datos que llegan al SCADA del cubesat, tanto los que son enviados por el propio cubesat como los que provienen del ADCS. Además al estar aplicado a un banco de pruebas de ADCS, se requería graficar en pantalla la orientación cubesat, por el medio de la creación de un cubo que se orientase según la actitud proveniente del ADCS. También se requería de una zona en la cual se imprimiese sobre un mapamundi la órbita del cubesat, en forma de traza terrestre. Por último se requería de un sistema de control que permitiese controlar la actitud del cubesat y la posición de este.

A partir de las condiciones anteriores, se ha diseñado el SCADA que se muestra en 14, el cual se aplicaría a un banco de pruebas de ADCS. Este SCADA está formado por seis módulos, los cuales se encuentran descritos a continuación. Cada módulo es desarrollado más a fondo en las sigueintes secciones 3.4 3.5 3.6 3.7 3.8. Todos los módulos se encuentran programados en distintas funciones, y estas se agrupan dentro de una clase llamada cubesat, lo que permite que el programa principal solo necesite crear un objeto de la clase cubesat, y llamar a sus funciones.

- A la izquierda del todo se ubican diez inputs de texto. Los dos de arriba se utilizan para introducir los valores de longitud y latitud de las estaciones terrestres que se quieren añadir al mapa terrestre. Al hacer click en 'ENVIAR' aparecerá una imagen de una estación terrestre sobre el mapa en la posición introducida. También hay ocho inputs que sirven para añadir la longitud y latitud de los vertices de la área de estudio. Al darle a 'ENVIAR' se creará un cuadrilátero en el mapa donde los vértices serán los añadidos previamente en los inputs de texto.

- A la derecha del elemento se encuentra el cuadro de control que permite aumentar y disminuir en uno los valores angulares y de posición.

- A la derecha de este, arriba, se encuentra un cuadro con los distintos datos generales del cubesat y sus valores actuales.

- A la derecha del elemento anterior se observa un cubo que simula ser el cubesat, colocado con los ejes de órbita, y los ejes del cubesat, para visualizar los ángulos.

- Abajo de los dos elementos anteriores se encuentra el mapa terrestre donde se muestra la trayectoria seguida por el cubesat en su órbita, y las estaciones terrestres y el área de estudio.

- Abajo del mapa se encuentra un cuadro con los datos de posición del cubesat, y la velocidad de este.

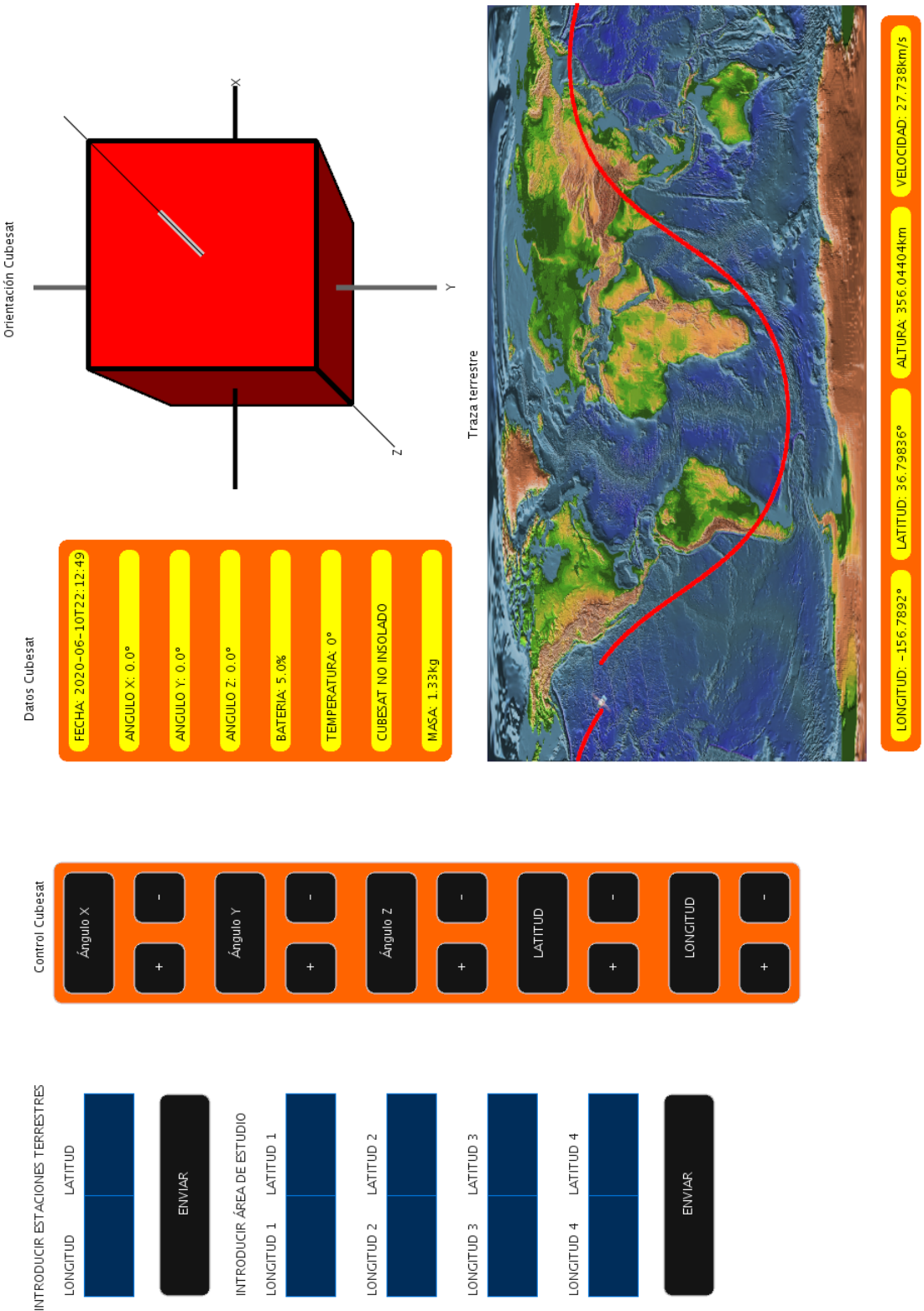

Figura 14: Diseño del SCADA del proyecto

3.4. Visualización de datos numéricamente

Esta parte del SCADA se encarga de la visualización de los datos en pantalla, pero en formato numérico. En este caso se refiere a los dos elementos que se muestran en pantalla con los valores numéricos ya procesados. Los datos generales del cubesat están separados de los datos de la órbita. Esto se debe a que así es más fácil diferenciarlos dado que los datos generales del cubesat se encuentran junto a la graficación 3D de este, y los datos de la órbita se encuentran debajo del mapa terrestre donde se enseña la posición del cubesat.

El primer elemento 15 hace referencia a los datos generales del cubesat y incluye los siguientes valores, los cuales han sido descritos en 3.2

- 'EPOCH´ o fecha en la que se envían los datos.
- Ángulo X
- Ángulo Y
- Ángulo Z
- Batería
- Temperatura
- Insolación
- Masa

Figura 15: Cuadro de datos generales del cubesat

El segundo elemento 16 hace referencia a los datos de órbita del cubesat y incluye los siguientes valores, explicados también previamente en 3.2

- Longitud
- Latitud
- Altura
- Velocidad

Figura 16: Cuadro de datos orbitales del cubesat

Estos dos elementos han sido programados en dos funciones diferentes, incluidas en la clase llamada cubesat. Estas dos funciones reciben el nombre de 'dataread' y de 'display'. La primera función se encarga de leer los datos del archivo de datos, procesarlos, cambiarles el formato a float para que el programa pueda interactuar con ellos y guardarlos en las variables creadas con el nombre de cada dato. La segunda función se encarga de acceder a estas variables que contienen los datos ya procesados, y enseñarlos por pantalla, junto con sus unidades.

3.5. Visualización 3D de la actitud

Este elemento del SCADA se encarga de la visualización 3D de los datos de actitud del cubesat. El elemento sería un cubo que se encuentra orientado según los ángulos que le llegan del archivo de datos, y según las órdenes que le llegan de los botones de control. En este caso el programa antes de dibujar el cubo se rota el plano el número de grados especificado para cada eje. Esto hace que los dibujos y imágenes mostrados a partir de ese instante, se encuentren en un plano diferente al principal, haciendo así que el cubo aparezca girado en pantalla.

La base de rotaciones utilizada se puede observar en 18, viendo que el sentido positivo de giro corresponde con el sentido positivo de los ejes.

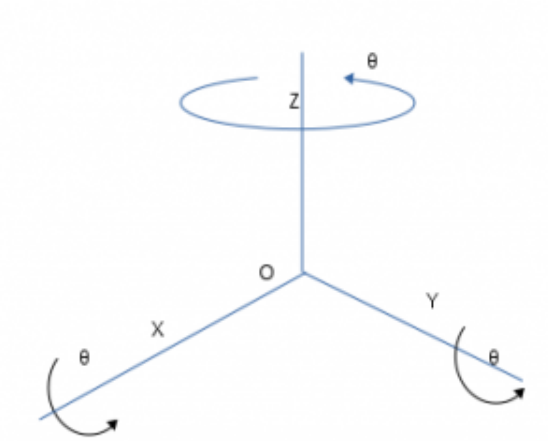

Figura 17: Base de rotaciones [15]

La orientación final del cubesat será la correspondiente a la suma de los tres giros alrededor de los tres ejes. Un ejemplo de un ángulo X se muestra en en 18, donde se observa un giro de una cantidad de grados α alrededor del eje X. Lo mismo aplicaría para los ángulos Y y Z, donde se giraría una cantidad de grados alrededor del eje Y o del eje Z.

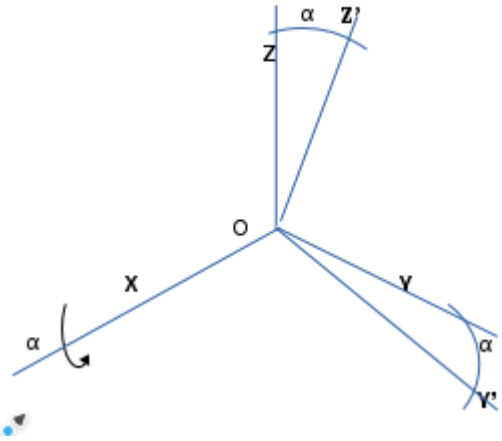

Figura 18: Ejemplo de ángulo X = α [15]

Como se puede ver en 19 en pantalla se muestran los ejes correspondientes a la órbita en línea negra fina, los cuales serían los ejes X,Y y Z de la figura 18. También se muestran los ejes correspondientes al cubesat, los ejes X', Y' y Z' de la figura 18, en una línea más gruesa. Además, para diferenciar las caras entre sí, dado que todas son lisas y del mismo color, el eje X' del cubesat se muestra en negro, el eje Y' se muestra en gris fuerte y el eje Z' se muestra en gris claro.

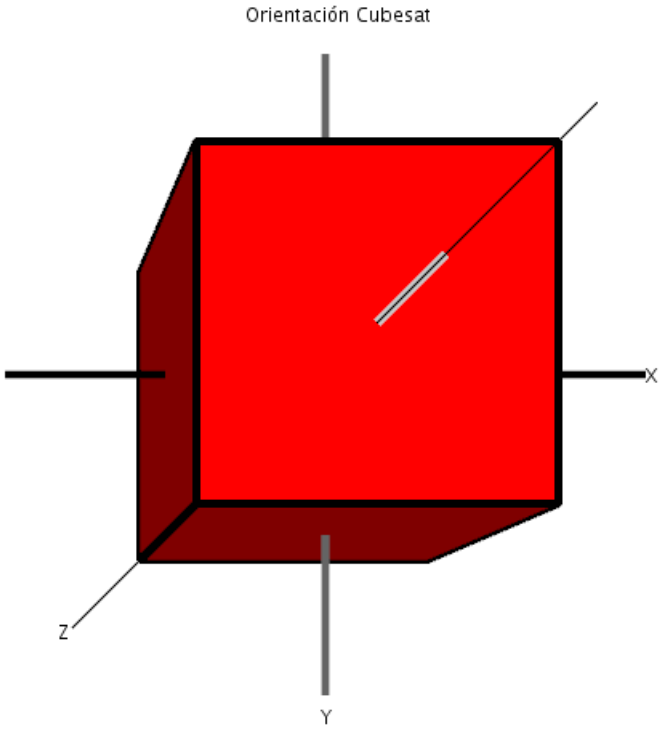

Figura 19: Visualización 3D de la actitud del cubesat

Este elemento ha sido programada dentro de la función "displayçomentada anteriormente en 3.4. Se encarga de leer las variables que contienen los datos de los ángulos, girar el plano estos ángulos, y de dibujar el cubo 3D en el nuevo plano.

Además, se incorpora una señal de alarma que cambia el color de rojo a amarillo en el caso de que el valor de la batería restante sea 5 % o inferior. Esto también se lleva a cabo en la función "display", mirando si el valor de la variable batería es inferior a 5 %, y cambiando el color del cubesat en caso de que esta sea inferior.

3.6. Visualización de la órbita

Este elemento del SCADA cumple la función de mostrar la órbita que sigue el cubesat. En este caso se imprime la traza terrestre sobre un mapa del mundo. En el caso de que se cambie la longitud y latitud del cubesat, se muestra la trayectoria que seguía antes, y se continúa con la nueva órbita. En la figura 21 se muestra la traza terrestre de la primera órbita del archivo de datos del cubesat.

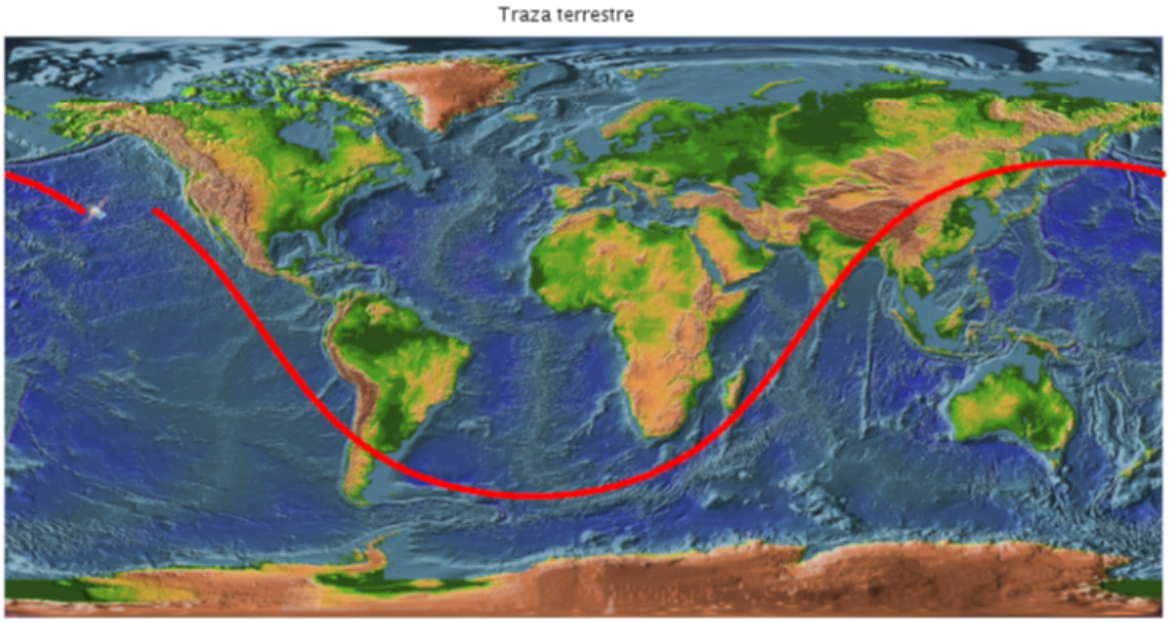

Figura 20: Traza terrestre del cubesat

Como se puede observar, el programa imprime la trayectoria que ha seguido el cubesat sobre el mapa terrestre. Esto se consigue mediante una reescala de los datos, y una transición de estos. La longitud y latitud se miden en dos ejes que van del -180º a 180º y de -90º a 90º, centrado en 0º, 0º. Esto se tiene que mostrar sobre el mapa, el cual tiene unas dimensiones de 750 de ancho y 376 de alto. Además el mapa está centrado en width/2-220, donde width corresponde al valor del ancho de la ventana, y height/2, donde height corresponde al valor del alto de la ventana. Se utilizan estos valores para que que al reescalar la ventana cuando se pone el SCADA eb modo pantalla completa no se muevan los elementos de posición. Después, se transforman los valores de latitud y longitud de una base a otra. Las fórmulas utilizadas para obtener los nuevos valores de posición se muestran a continuación. La latitud se le cambia el signo dado que como se muestra en 12 los valores de latitud aumentan cuanto más arriba del mapa se encuentren, mientras que Processing sitúa el inicio de coordenadas en el borde superior izquierdo de la pantalla, por lo que el valor de altura aumentará cuanto más abajo nos encontremos de la pantalla.

$$Posicionx = longitud/180 \cdot 375 + width/2 + 155 \tag{1}$$

$$Posiciony = -latitud/90 \cdot 188 + height/2 + 188 \tag{2}$$

Estos nuevos valores de posición obtenidos de 1 y 2 corresponden a la posición de la pantalla donde se dibujará un pequeño cuadrado para la traza terrestre, y se mostrará la imagen de un satélite para la posición actual. Además para acceder a todos los valores hasta el momento, el programa utiliza el comando for, accediendo en cada iteración a los valores de latitud y longitud del archivo de valores desde le inicio hasta la línea en la que se encuentra. Este elemento está programado dentro de la función .ºrbita", la cual se encarga de procesar los datos de latitud y longitud, y dibujarlos en el mapa terrestre.

3.7. Control

La parte de control de un SCADA es importante para un banco de pruebas de ADCS, dado que envía las órdenes y nuevos datos al procesador de ADCS. Como en este caso no se contaba con un banco de pruebas ni un ADCS, se ha simulado la actuación de este mediante la creación de variables secundarias. Estas permiten controlar la actitud y la posición del cubesat. En **??** se muestra la interfaz de control diseñada para el SCADA. Esta interfaz permite controlar:

- Ángulo X
- Ángulo Y
- Ángulo Z
- Longitud
- Latitud

Figura 21: Interfaz de control del SCADA

Como se observa en 21, el interfaz de control permite aumentar y disminuir los ángulos del cubesat, pero solamente 1º. Esto se lleva a cabo mediante variables de control creadas para simular la actuación del hardware de control del ADCS. Al hacer click sobre el botón '+' se aumenta en uno la variable correspondiente, y al hacer click sobre el botón '-' se disminuye en uno el valor de dicha variable. Acto seguido, el valor leído del archivo de datos se suma con el valor de la variable de control que le corresponda. Esto hace que el control permita interactuar con la simulación 3D de la actitud del cubesat. Si solo se aplicase la variable de control en el instante en el que se presiona el botón y después se mostrasen los datos del archivo sin alterar, habría una discontinuidad en el programa, ya que solo se controlaría un instante de tiempo y después volvería a la normalidad.

Para simular la actuación del hardware de control de ADCS, y que este afecta también a los datos futuros, la variable se suma a partir del momento en el que se hace click en el botón, y se sigue sumando durante todos los siguientes valores de datos.Esto permite que se pueda controlar la actitud de la simulación del ADCS, mientras este sigue el modelo de los datos del archivo de datos. La actitud final del satélite se calcula con al fórmula 3

$$Valorfinall = valordelarchivodedatos + variabledecontrol \tag{3}$$

Esto hace que la simulación siga el modelo del archivo de datos, dado que la variable de control se mantiene constante a no ser que se presione los botones. Cuando se hace click en los botones, esta variable se aumenta o disminuye, lo que hace que el angulo enseñado cambie. Esto permite girar el cubesat siguiendo la base de rotaciones que se muestra en 18, donde el botón '+' crea giros en sentido positivo, y el botón '-' crea giros en sentido negrativo. En la imagen **??** se muestra como se ha cambiado la actitud del cubesat, tomando habiendo controla el cubesat para que cambiase de la posición de 19 donde todos los ángulos son 0º, a la posición de 45º en todos los ejes.

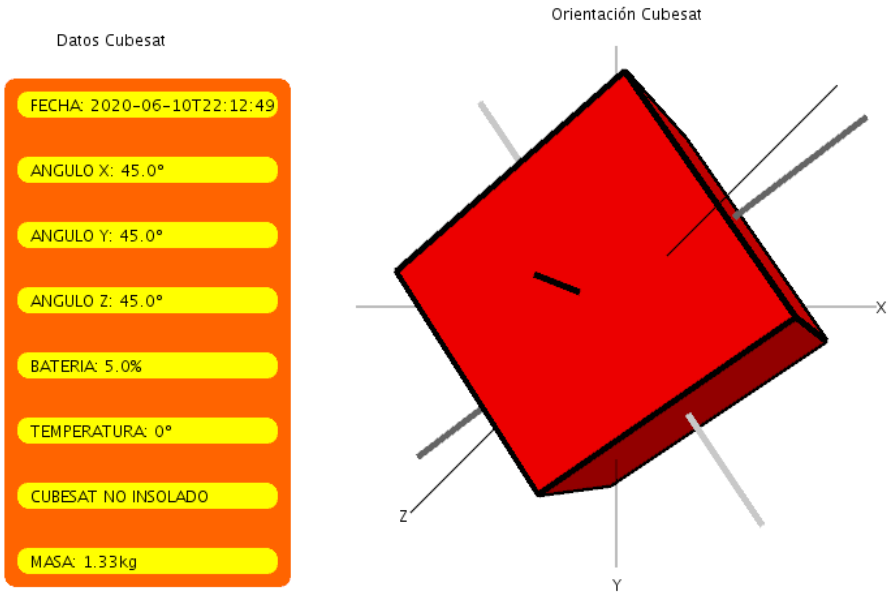

Figura 22: Ejemplo de control de la actitud

Como se observa en el interfaz de datos numéricos, los tres ángulos son de 45º. Estos se han logrado enviando señales de control a la simulación, y variando la simulación hasta que esta coincide con los datos requeridos. En otras palabras, no se puede disminuir de golpe un valor de 45º el cubesat, sino que se tendrá que disminuir de uno en uno hasta llegar al valor requerido. Esto parece poco eficiente, pero el loop de Processing hace que no sea necesario hacer click 45 veces, sino que cuanto más rato se deje apretado el ratón sobre el botón de '+' más se aumenta el valor de la variable de control.

Este diseño también tiene ventajas, ya que en el caso de contar con un cubesat y banco de pruebas, esto permitiría controlar la posición del cubesat de manera eficiente, mirando los datos que se muestran en el interfaz numérico hasta obtener el valor final deseado. Después se puede comparar la actitud de la simulación 3D y como esta ha evolucionado con el tiempo con la actitud que se observa en el cubesat del banco de pruebas, para ver si el cubesat real ha seguido el modelo del virtual y que ambos se encuentran en la misma orientación. y por lo tanto verificando que el hardware y el procesador del ADCS funcionan correctamente y que los tests son satisfactorios.

La figura 21 también muestra que se permite controlar la latitud y longitud del cubesat. Estos botones funcionan igual que los de control de la actitud, y aumentan y disminuyen en 1º la longitud o la latitud del cubesat. Los valores de las variables de control también se añaden a los valores del archivo de datos siguiendo la fórmula 3. En este caso, funciona diferente dado que se necesitan mostrar la trayectoria antes del momento en el que esta cambia intacta. Esto se consigue creando un array que contienen el valor de la variable de control de la longitud y la latitud para cada valor de longitud y latitud del archivo de datos. En un inicio, se va guardando siempre el mismo valor en cada posición del array, pero en el momento en el que se presiona el botón, los valores del array que se guardan aumentan o disminuyen en 1, sin cambiar los valores de las posiciones anteriores. Esto hace que al llamar a todos los valores utilizando un for, cada valor de longitud y latitud tenga un diferente valor dependiendo de si se ha presionado los botones de control en ese instante o no. A partir del momento de modificación de la órbita, el valor que se guarda en el array pasa a ser el mismo, es decir, que la variable de control lo que hace es desplazar la órbita a partir del momento en el que se hace click, la determinada cantidad de grados que ponga en el array.

Un ejemplo sería 23 donde se aumenta la latitud y longitud del SCADA. Esto se puede ver comparado con 21, que corresponde a la órbita sin aplicar ningún control sobre ella. La primera acción que se hace es aumentar la latitud. En este caso no se muestra como un ascenso vertical dado que la longitud sigue aumentando al ritmo que marca el archivo de datos porque que sobre esta no se ha aplicado ningún control, y por tanto se muestra como un aumento en diagonal. Después se encuentra el aumento de la longitud, mostrado como una parte recta en medio de la órbita. Este aumento, como el anterior, es en diagonal porque se aumenta solamente la longitud del cubesat, pero la latitud sigue disminuyendo al ritmo del archivo de datos. Al final el cubesat sigue la nueva órbita, que si se compara con la anterior, se puede ver que se encuentra desplazada hacia arriba y hacia la derecha, debido a que la latitud y longitud de estas han aumentado.

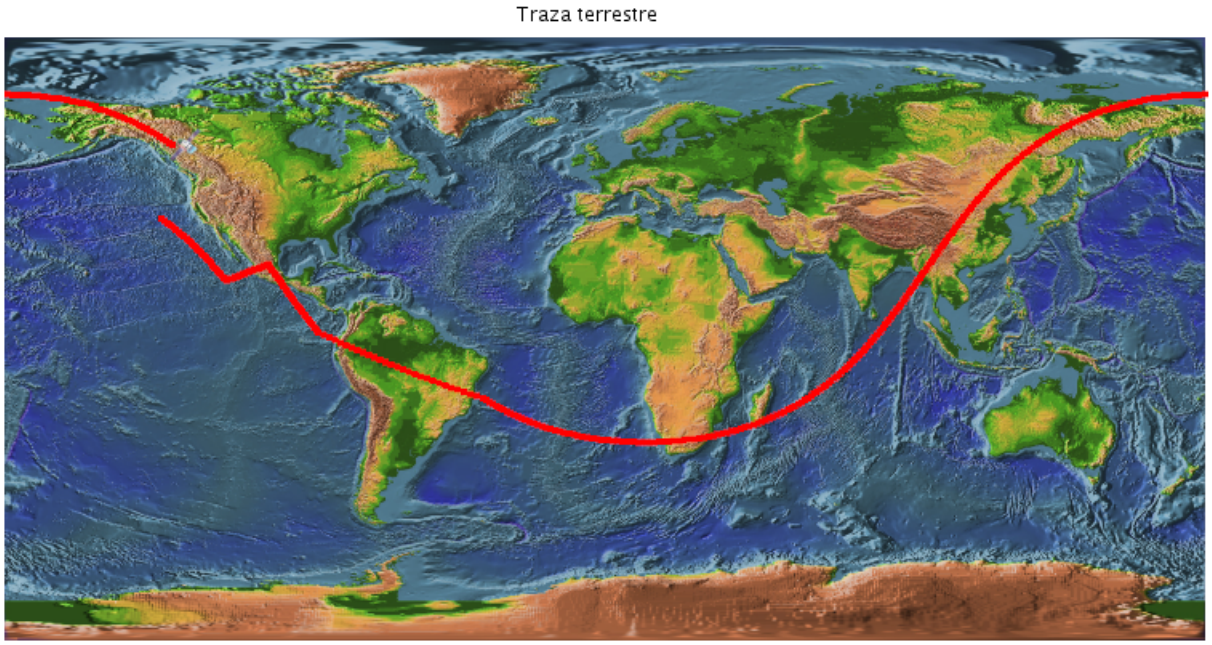

Figura 23: Ejemplo de interacción con la órbita

El problema de calcular el valor de la longitud y la latitud utilizando la fórmula 3 es que cuando la longitud es superior a 180º o inferior a -180º, y la latitud es superior a 90º o inferior a -90 se dibuja fuera del mapa terrestre del SCADA. Entonces se han tenido que limitar los valores de longitud y latitud totales. Si no se hubiera programado el límite de estos, se habría obtenido un resultado parecido al de la figura 24, donde la órbita se dibuja fuera del mapa.

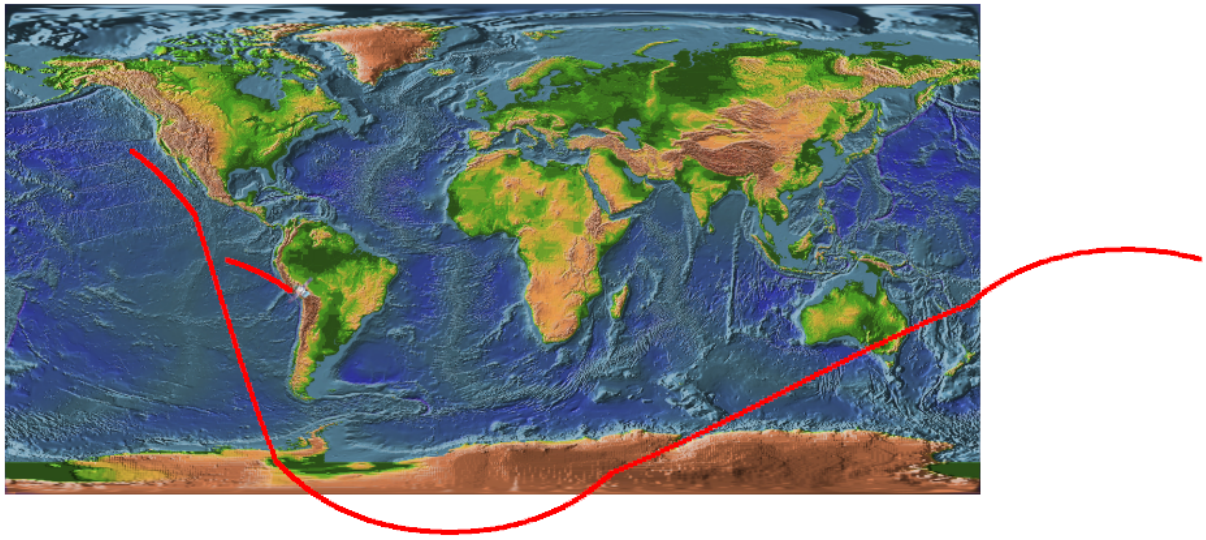

Figura 24: Error al no limitar la longitud y la latitud

Esto se soluciona cambiando el valor de la variable de control. Así, cuando una vez la longitud total y la latitud total llegan al límite, el valor de la variable de control cambie, para que se mantenga la latitud y la longitud total cambie a la contraria, o para que se mantenga la longitud y la latitud total cambie a la contraria. Las fórmulas utilizadas para calcular los nuevos valores de las variables de control se muestran en 4, 5, 6, 7 y 8. Estas fórmulas son diferentes para cuando los valores del archivo de datos de longitud sean positivos o negativos, dado que en este caso en el archivo de datos se pasa de tener valores positivos a negativos de longitud, y si no se varía la fórmula habría una discontinuidad en la traza terrestre, dado que hay un cambio en el signo de la longitud del archivo de datos, y si se suma tal cual con la variable de control con el valor anterior, daría un valor inferior a -180º, dado que sería sumar un número negativo a -180º. En otras palabras, lo que se hace cuando el valor de longitud o latitud se sale de los límites es cambiar variable de control para que la órbita se traslade al lado opuesto del mapa. Excepto en el caso de que el valor de la longitud del archivo de datos cambie, que lo que se hace es cambiar la variable de control para que el satélite se mantenga en el mismo punto.

- Si la longitud total es más grande de 180º :

$$variabledecontrollongitud = -180 - longituddelarchivodedatos \tag{4}$$

- Si la longitud es más pequeña de 180º y el valor de la longitud del archivo de datos tiene el mismo signo que el anterior:

$$variabledecontrollongitud = 180 - longituddelarchivodedatos \tag{5}$$

- Si la longitud es más pequeña de 180º y el valor de la longitud del archivo de datos tiene diferente signo que el anterior:

$$variabledecontrollongitud = longitudtotalanterior - longituddelarchivodedatos \tag{6}$$

- Si la latitud es más grande que 90º:

$$variabledecontrollatitud = -90 - latituddelarchivodedatos \tag{7}$$

- Si la latitud es más pequeña que 90º:

$$variabledecontrollatitud = 90 - latituddelarchivodedatos \tag{8}$$

A partir de estas fórmulas, que varían las variables de control, se consigue que se pueda controlar el cubesat y que este siempre se encuentre dibujado dentro de los límites del mapa. Un ejemplo se observa en la figura 25 donde se aumenta la longitud y se disminuye la latitud para que sus valores totales sean mayores que los límites, y como se dibujan al lado opuesto del mapa siguiendo la órbita del archivo de datos cuando esto sucede.

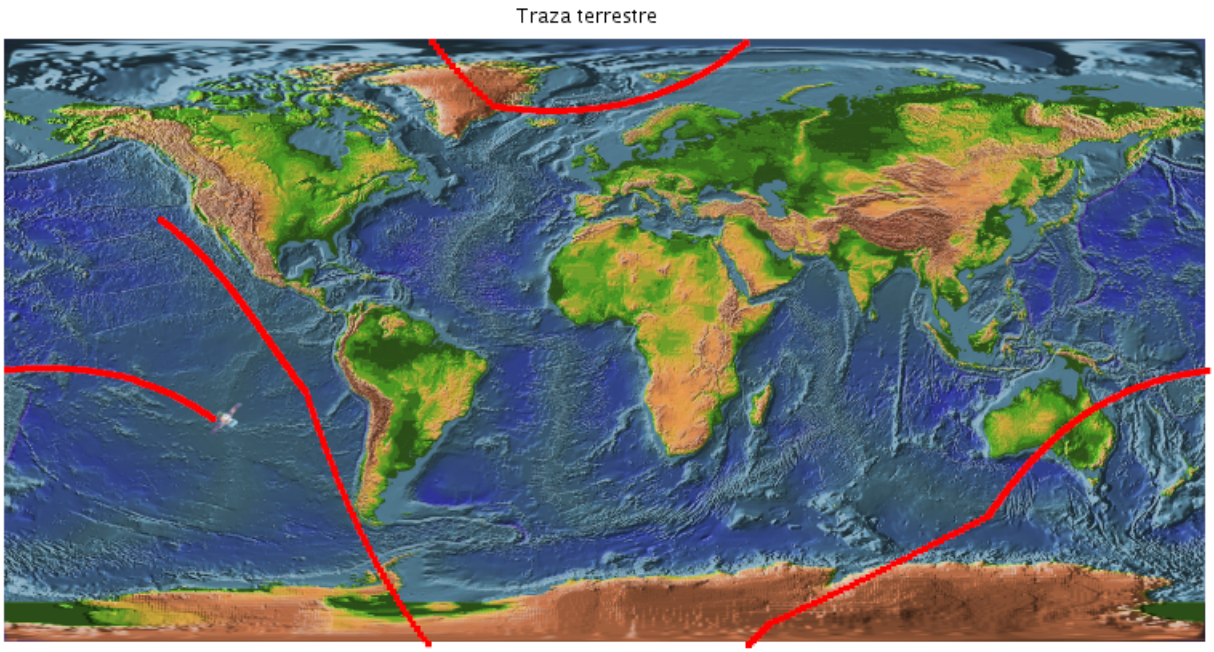

Figura 25: Ejemplo donde la latitud y longitud cambian de signo

En la figura 25 se puede observar como se disminuye la latitud hasta que esta es menor que -90º, y entonces su valor pasa a ser +90º, y como la longitud se aumenta y cuando su valor es mayor de 180º, este pasa a ser -180º.

3.8. Input

Este elemento del SCADA es una mejora añadida al entorno, que sirve para visualizar las estaciones terrestres más cercanas y las zonas de estudio del cubesat. Esto se consigue mediante dos inputs de texto, para la longitud y latitud de la estación espacial, y ocho inputs, para la longitud y latitud de los cuatro vértices del cuadrilátero que encierra el área de estudio, como se muestra en 26. Los datos de la estación terrestre que se quiera añadir se tienen que introducir a la vez, y después hacer click en el botón de 'ENVIAR' para que estos se envíen al programa. Lo mismo sucede con la longitud y latitud de los cuatro vértices, que una vez estén todas escritas hay que hacer click en el botón de 'ENVIAR' inferior para enviarlos. Una condición añadida es que los vértices se tienen que introducir siguiendo en sentido horario o antihorario los vértices cuadrilátero. Estas dos condiciones son importantes, dado que sino el programa deja de funcionar ya que no puede procesar los datos introducidos si el espacio está en blanco, y también porque el cuadrilátero se dibujará mal si no se ponen los vértices siguiendo un orden horario o anithorario.

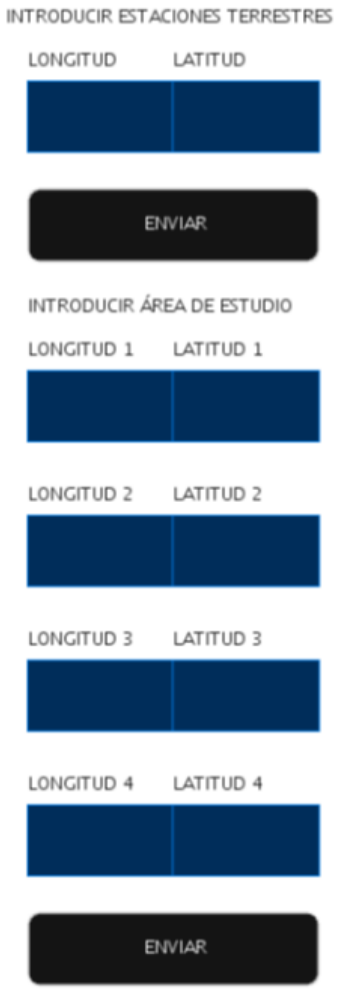

Figura 26: Cuadros de texto donde se introducen los datos y botones para enviarlos

Una vez introducidos los valores de latitud y longitud de las estaciones terrestres o del área de estudio, su posición en pantalla es recalculada utilizando las equaciones 1 y 2 dado que los datos que se introducen son latitud y longitud igual que para la traza terrestre. La latitud y longitud son guardados en arrays de datos, en vez de variables únicas, para que así sea posible introducir más de una estación terrestre y más de un área de estudio. El programa es parecido al de la traza terrestre, y se utiliza un for para leer todos los valores de los array de latitud y longitud. Cada vez que se accede a un valor del array, se imprime un dibujo de una antena en la posición donde se encuentre la estación terrestre. En el caso de la área de estudio, se dibuja un cuadrilátero blanco que delimita el área de estudio.

Un ejemplo se puede observar en la figura 27 donde se han introducido dos estaciones espaciales en las coordenadas de Barcelona y de Nueva York. Además se han limitado dos áreas de estudio, una cuadrada y otra irregular.

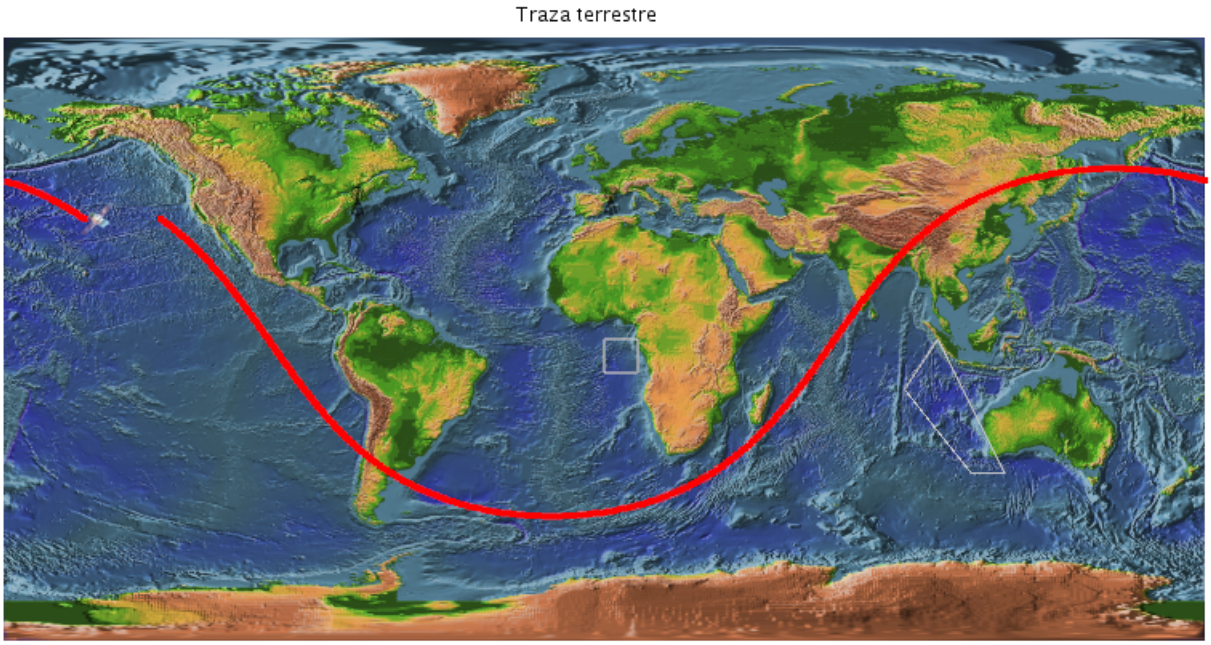

Figura 27: Ejemplo con dos estaciones terrestres y dos áreas de estudio

Las dos imágenes de antenas corresponden a las posiciones de las estaciones terrestres introducidas, y los cuadriláteros de color gris claro corresponden a las dos áreas de estudio introducidas.

3.9. Diagrama de flujo

Una vez se han explicado los elementos que forman el SCADA, se procede a mostrar el diagrama de flujo de este. El diagrama de flujo muestra los pasos que sigue el SCADA desde su inicio hasta su final. En este caso como Processing utiliza un loop infinito, el SCADA no tiene final y vuelve al inicio todo el rato, actualizando los datos del archivo de datos. Primero se carga el archivo de datos y después se procesan los datos obtenidos. Seguidamente se añaden las variables de control de órbita a los datos de órbita, se comprueba si los datos de longitud y latitud obtenidos están dentro de los límites y se dibuja la órbita. Después se suman las variables de control de actitud con los datos de actitud y se dibujan por pantalla tanto numérica como gráficamente. Después se dibujan los controles y en el caso de que alguno se presione, se varía la variable de control correspondiente. Seguidamente se pueden introducir manualmente los datos de posición de las zonas terrestres o se puede pasar a la siguiente fase en el caso de no introducir ningún valor. Estos valores solo le guardan si se presiona el botón de 'enviar'. Después se lleva a cabo una acción igual a la anterior pero con los datos de posición de las áreas de estudio, donde o se introducen los datos de posición del área de estudio, los cuales solo se guardan si se presiona el botón de 'enviar', o se pasa a la siguiente acción. Finalmente se dibujan las estaciones terrestres y las zonas de estudio guardadas durante todo el programa. Al acabar esta acción se vuelve al inicio, repitiendo el ciclo nuevamente para una nueva línea de datos.

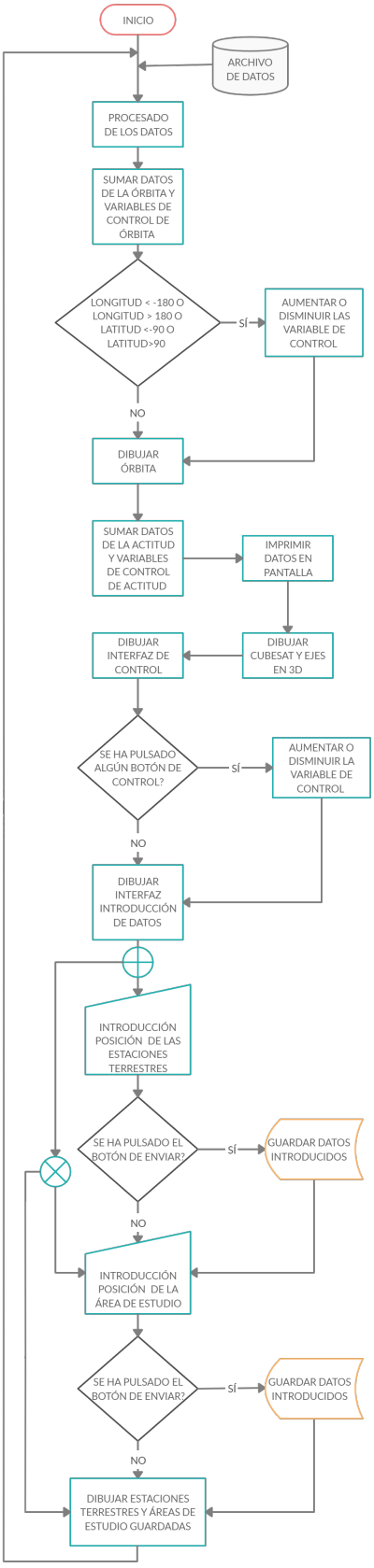

Figura 28: Diagrama de flujo del SCADA

4. Presupuesto

En este apartado se explica la planificación y programación que se ha seguido durante la realización del proyecto junto con las horas dedicadas a este. También se muestra el presupuesto del proyecto y las implicaciones ambientales de este.

4.1. Planificación y programación

En esta sección se muestran las horas que se han dedicado a cada tarea del proyecto, y la planificación que se ha seguido durante el proyecto. La figura 29 muestra las tareas que se han llevado a cabo, y la distribución de estas a lo largo de las semanas.

TAREAS	Sem1 17/02-23/02	Sem2 24/02-01/03	Sem3 02/03-08/03	Sem4 09/03-15/03	Sem5 16/03-22/03	Sem6 23/03-29/03	Sem7 30/03-05/04	Sem8 06/04-12/04	Sem9 13/04-19/04	Sem10 20/04-26/04	Sem11 27/04-03/05	Sem12 04/05-10/05	Sem13 11/05-17/05	Sem14 18/05-24/05	Sem15 25/05-31/05	Sem16 01/06-07/06	Sem17 08/06-14/06	Sem18 15/06-21/06	Sem19 22/06-28/06	Sem20 28/06-05/07
PROJECT CHARTER																				
REUNIONES																				
MEMORIA																				
BÚSQUEDA SOBRE CUBESATS																				
BÚSQUEDA SOBRE SCADA																				
TUTORIALES EN PROCESSING																				
VISUALIZACIÓN DE DATOS																				
VISUALIZACIÓN CUBESAT																				
CONTROL CUBESAT																				
VISUALIZACIÓN ÓRBITA																				
ENTORNO FINAL																				

Figura 29: Planificación de las tareas a lo largo de las semanas del proyecto

- La primera tarea es la redacción del project charter. Esta se lleva a cabo al inicio del proyecto, durante tres semanas.
- Las reuniones con el tutor, al principio eran reuniones presenciales semanales con el tutor, para aclarar el proyecto y empezar con su desarrollo. Debido al coronavirus, estas han pasado a ser online y de control cada tres semanas.
- La redacción de la memoria empieza en el día que se entrega el project charter, 30 de Abril, y continúa semanalmente.
- La búsqueda sobre cubesats se centra principalmente al inicio del proyecto, aunque después también se hacen búsquedas semanales sobre ciertos temas concretos cuando se está programando el control y la visualización de la órbita.
- La búsqueda sobre SCADA sobre se lleva a cabo al inicio del proyecto, y a partir de la información obtenida se diseña un entorno específico para el proyecto
- La visualización y réplica de tutoriales en Processing para aprender el lenguaje y funcionamiento del entorno se lleva a cabo al inicio del proyecto, pero también cada vez que se programa un nuevo elemento del SCADA, dado que incorporan nuevas funciones que se tienen que aprender a programar.
- La programación del código de visualización de datos, explicado en 3.4, se lleva a cabo al inicio y es la que más dura dado que no se tiene mucho conocimiento del proyecto y se

tienen que hacer varios modelos y pruebas junto con tutoriales para aprender a utilizar Processing. También durante la fase final, se retoca para que todo cuadre en conjunto

- La programación del código de visualización del cubesat, que corresponde al apartado 3.5, se lleva a cabo a la vez que el anterior, y por lo tanto tienen la misma duración.
- La programación del sistema de control del cubesat, que se explica en 3.7, se lleva a cabo después de las anteriores, y se refiere a la parte de control de los dos entornos anteriores. Después hay una fase que se lleva a cabo con la programación del código de visualización de órbita dado que este también se tiene que controlar con la interfaz de control, y no se puede programar el control de este hasta no tener desarrollada la visualización de órbita.
- La visualización de la órbita, explicado en 3.6, es el último elemento que se programa por separado, y se retoca al inicio de la programación del entorno final para que todo cuadre.
- La última tarea es la programación del entorno final, donde se retocan todos los entornos programados anteriormente para que tengan las unidades correctas y para que se muestren conjuntamente en pantalla, además se programa el código explicado en 3.8, que permite introducir la latitud y longitud de las estaciones terrestres y de la zona de estudio.

En la siguiente tabla 3 se muestran el total de horas dedicada a cada tarea, y las horas totales dedicadas al proyecto.

Tarea	**Horas invertidas**
Redacción project charter	20
Reuniones con el tutor	8
Redacción Memoria	95
Búsqueda cubesats	55
Búsqueda SCADA	40
Tutoriales	80
Visualización datos	65
Visualización cubesat	65
Control cubesat	80
Visualización órbita	60
Entornofinal	55
Total de horas	**623**

Tabla 3: Horas dedicadas a cada tarea del proyecto

4.2. Presupuesto

En este apartado se detalla el coste del proyecto. En este caso no se trata de un proyecto muy costoso, dado que no se cuenta con el banco de pruebas ni con el ADCS, por lo que solo se tiene en cuenta la energía gastada y las horas invertidas en su realización. El cálculo del coste de la electricidad se ha hecho de la siguiente manera:

$$Ce = 0,200kW \cdot 618h \cdot 0,13/kWh = 16,07 \tag{9}$$

Donde Ce es el coste de la electricidad, los 0,2kW es la energía que consume el ordenador, 618h es el tiempo que se ha estado utilizando, dado que se quitan las reuniones presenciales con el tutor, y 0,13 €/kWh es el coste de la electricidad según [16].

El gasto de mano de obra se ha calculado de la siguiente forma:

$$Cmo = 623h \cdot 15/h = 9345 \tag{10}$$

Donde Cmo es el coste de la mano de obra, 623h son las horas que se ha trabajado en el proyecto, y 15 €/h es el salario escogido para el trabajador.

El total entonces se calcula siguiendo la fórmula 11:

$$Ct = Ce + Cmo = 16,07 + 9345 = 9361,07 \tag{11}$$

Por lo tanto, el coste total del proyecto sería de 9361€, que equivaldría a la suma del coste de la mano de obra más el coste de la electricidad.

4.3. Implicaciones ambientales

En esta sección se muestran las implicaciones ambientales del proyecto. En este caso solo se tiene en cuenta las emisiones de C02 que se han creado con la producción de energía para la alimentación del ordenador que se ha utilizado en el proyecto. En el caso de que se contase con el banco de pruebas y del ADCS, se le tendría que añadir a este valor el consumo energético de ambos, y las horas de trabajo que se han tenido ambos funcionando en el taller.

La media de emisión de C02 creada para generar un kW de energía es de 0,372 kg/kWh según [17]. Cogiendo este dato y multiplicándolo por la alimentación del ordenador y las horas de funcionamiento, como se muestra en 12, se obtiene la cantidad de C02 emitido para llevar a cabo este trabajo.

$$Emisiones = 0,200kW \cdot 618h \cdot 0,372kg/kWh = 45,98kgdeCO2 \quad (12)$$

La cantidad total estimada de CO2 que se ha emitido a la atmósfera en la realización de este proyecto es de 45,98 kg. Si se comparan estas emisiones a las emisiones de un coche de gasolina, estas son las que provocaría un coche al recorrer 280km, extraído de [18]. Esto muestra que las emisiones de CO2 no son muy elevadas, dado que un coche puede hacer los 280km en dos o tres días, mientras que las emisiones de CO2 del proyecto se han hecho a lo largo de las 20 semanas de duración de este.

5. Conclusiones y desarrollo futuro

Como se explica en 1.1, el objetivo de este proyecto es desarrollar un entorno de control y visualización de un banco de pruebas de ADCS. En esta memoria se comentan los pasos que se han seguido para completar este objetivo, además que se explica el entorno final SCADA diseñado. Debido a que no se disponía de un banco de pruebas ni de un ADCS, se han tenido que implementar cambios en el proyecto, como la existencia de un archivo de datos o unas variables de control, dado que no se puede conectar el programa con un ADCS. Estos cambios también están incluidos en el contenido de esta memoria.

Los resultados del proyecto han sido satisfactorios, dado que el entorno SCADA diseñado en este proyecto cumple los objetivos de permitir visualizar los datos y de poder controlar al cubesat a través de los elementos de control de este.Además, los cambios realizados al proyecto son satisfactorios, y tanto el archivo de datos como las variables de control simulan correctamente la existencia de un banco de pruebas y de un ADCS. El SCADA muestra la órbita que sigue el satélite y permite controlarla des de la interfaz de control. Por lo tanto, se puede decir que los requisitos del proyecto han sido cumplidos con éxito. Además, se cumple con el alcance del proyecto para cada elemento. Por último, se ha añadido al proyecto una mejora, la posibilidad de introducción de la posición de las estaciones terrestres y del área de estudio en unos cuadro de texto, para que estas se muestren en pantalla impresas sobre el mapa del SCADA.

Al código se le pueden implementar algunas modificaciones para mejorar su funcionamiento en trabajos futuros. Una modificación sería que al introducir la latitud y longitud de la área de estudio, las coordenadas se ordenasen solas. Esto haría que no hiciese falta introducir los vértices en orden horario o antihorario, dado que el área se dibujaría correctamente aunque se introdujesen los vértices en un orden aleatorio. Otro cambio sería hacer un sistema que se encarga de comprobar si los datos introducidos en los inputs de textos son números, y en el caso de no serlo, que diese una señal de error, para que el usuario cambie las coordenadas introducidas. Esto haría que el programa no se colgase cuando se introducen palabras o se dejan en blanco los espacios en los inputs de textos como ocurre ahora. Por último, se podrían reprogramar los botones de control, y poner inputs, donde en vez de aumentar los datos de 1º en 1º, se podría introducir directamente el valor que se quiere. Esto ayudaría a hacer más fluido el control, a la vez que la introducción de datos se llevaría a cabo de una forma más rápida.

Aparte, el desarrollo futuro de este proyecto es obvio, dado que en el momento en el que se cuente con un banco de pruebas y un ADCS, se le pueden aplicar mejoras. En este caso, para el futuro del proyecto, se tendrían que llevar a cabo tres tareas muy importantes.

- La primera tarea es la sustitución de la manera en la que le llegan los datos al SCADA. Se tendría que suprimir la existencia del archivo de datos, y programar el SCADA para que los datos lleguen vía serial y sean los medidos por el ADCS. También se tendrían que conectar los sensores con procesadores, como los del ADCS o una placa Arduino, para que se procesen las medidas que consiguen los sensores, y estas sean transformadas en datos que el SCADA puede leer y trabajar con ellos. Esto se podría llevar a cabo modificando la función 'dataread' de A. Se tendría que cambiar el método de introducción de datos, utilizando la librería processing.serial, la cual permite obtener datos a través de una vía

serial. Después se tendrían que leer los datos que llegan a través del puerto serial, en vez de leerlos a partir de un documento .csv. Por último se tendrían que guardar estos datos obtenidos vía serial en un array de datos que contuviese todos los datos que le han llegado durante su tiempo de funcionamiento.

- La segunda tarea que se tendría que llevar a cabo es la sustitución de las variables de control, por el procesador ADCS. En este caso, se tendría que programar que cuando se activan los elementos de control de la pantalla, se envíe una señal serial al procesador del ADCS. Este procesador transformaría la señal de entrada en órdenes para el hardware del SCADA, con el fin de conseguir en el cubesat los valores de actitud introducidos en el SCADA. La modificación del código se tendría que llevar a cabo en la función 'control' de A. Esta, en vez de actuar sobre las variables de control, utilizarían también la librería processing.serial, para conectarse con el procesador del ADCS, y enviarle por vía serial señales cada vez que se active alguno de los elementos de control. Cada botón se tendría que programar para enviar una señal diferente, así el procesador podría diferenciar entre qué botón se ha pulsado. Por último, el procesador obtendría las señales, y las transformaría en órdenes de control que se mandarían al hardware de control del ADCS.

- También se tendría que pensar en el control de la órbita del cubesat. Se tendría que realizar un estudio sobre el método que se quiere para controlar dicha órbita. Una opción serían los 'cold gas thrusters', donde se tendría que elegir el tipo de combustible que se quiere utilizar. Además, se tendría que estudiar la colocación de los thrusters en el cubesat. Por último, se tendría que utilizar un procesador, donde se le programe un algoritmo que envíe órdenes de control a los thrusters a partir de los datos de órbita que le llegan, para poner el satélite en la longitud y latitud requeridas. Estas modificaciones se tendrían que introducir también en la función 'control' de A, y serían unas modificaciones similares a las anteriores. Se conectaría por vía serial el SCADA con el procesador del hardware de control de órbita. Luego se tendrían que enviar señales cada vez que se active uno de los elementos de control de órbita, y el procesador transformaría estas señales en órdenes para los elementos de control de órbita.

Al implementarse estos cambios al proyecto, la arquitectura del SCADA pasaría de ser la mostrada en 10 a la arquitectura mostrada en 9. Sin embargo el desarrollo futuro de este proyecto solo se puede llevar a cabo si se dispone de un banco de pruebas de ADCS o de un ADCS. Estos cambios se pueden ir implementado gradualmente, y no necesitan implementarse todos a al vez. Por ejemplo en el caso de disponer de un ADCS, se puede implementar solamente la mejora del sistema de control de la actitud. Después cuando se disponga de un banco de pruebas de ADCS, se puede implementar la mejora en los datos de entrada. Y por último, una vez hecho el estudio de la propulsión y los algoritmos de control de órbita, se puede implementar la mejora del sistema de control de la órbita.

Agradecimientos

El proyecto presentado en este documento no podría haberse llevado a cabo sin la participación del director David González, quien ha servido de guía durante el proyecto, sobretodo al inicio de este, ayudando a esclarecer el diseño inicial del SCADA y resolviendo las dudas que se presentaban durante el desarrollo del proyecto. También agradecer la participación del codirector Ignasi Esquerra, quien ha ayudado a resolver dudas sobre Processing, y quien propuso la programación del sistema SCADA como un objeto, en lugar de utilizar funciones programadas en el programa principal. También agradecer al canal de youtube 'The coding Train', dado que al inicio sus tutoriales ayudaron a entender mejor el funcionamiento de Processing cuando nunca se había trabajado con este. Agradecimientos también a mi familia quienes me han animado a seguir adelante y progresar con el proyecto, sobretodo en momentos de frustación cuando un código no funcionaba o los resultados no eran satisfactorios. Por último agradecer a David Poza la gran ayuda aportada en Latex, dado que ha ayudado a resolver las dudas que iban apareciendo durante la redacción de la memoria y ha ayudado a resolver fallos en el código del documento en Latex.

Bibliografia

[1] Robert Bedington, Xueliang Bai, Edward Truong-Cao, Yue Chuan Tan, Kadir Durak, Aitor Villar Zafra, James A Grveve, Daniel Oi, Alexander Ling, *Nanosatellite experiments to enable future space-based QKD mission*, EPJQuantum Technology, National University of Singapore, Diciembre 2016.

[2] Herbert J. Kramer, *RaInCube (Radar In a CubeSat) - A Precipitation Profiling Mission*, eoPortal, 2018.

[3] Jyan Gu, Han, Congying, *WHERE IS THE LIMIT: THE ANALYSIS OF CUBESAT ADCS PERFORMANCE*, TU Delft Space Institute/Faculty of Aerospace Engineering, Delft University of Technology, Mayo 2016.

[4] Martin Langer, Florian Schummer, Nicolas Appel, Thomas Gruebler,Katja Janzer, Jonis Kiesbye, Lucas Krempel, Alexander Lill, DavidMessmann, Sebastian Rueckerl, Michael Weisgerber *MOVE-II THE MUNICH ORBITAL VERIFICATION EXPERIMENT II*, Institute of Astronautics, Technical University of Munich, Dieciembre 2017.

[5] NanoSat Lab, *Helmholtz Coil System*, UPC, nanosatlab.upc.edu, 2016.

[6] J. L. Schwartz, M. M. A. Peck y C. D. Hall, *Historical Review of Air-Bearing Spacecraft Simulators*, Journal of Guidance Control and Dynamics - J GUID CONTROL DYNAM, 2003.

[7] J. Molina, G. Bisiacchi, L. Reyes, E. Vicente, F. Contreras y M. Mesinas, *Threeaxis air-bearing based platform for small satellite attitude determination and control simulation*, Journal of Applied Research and Technology, 2005.

[8] REVERE, *SCADA BASICS: WHAT ARE SCADA AND TELEMETRY?*,Reverecontrol.com, Junio 2019.

[9] Kyle W Doerksen, Jean-François Lévesque, Ignacio Más, *Small Satellites Design , Modeling and Evaluation of a 2 . 4 GHz FHSS Communications System for NarcisSat*, Space Systems Development Lab Stanford University, Conference on Small Satellites, 2003.

[10] ANDREW HENRY, *WHAT IS OPEN MCT?*, nasa.github.io, 2020

[11] CCSDS, *ORBIT DATAMESSAGES*,CCSDS 502.0-B-2, Recommended Standard, Issue 2, Noviembre 2009.

[12] CCSDS, *NAVIGATION DATA— DEFINITIONS AND CONVENTIONS* ,CCSDS 500.0-G-3, Informational Report, Issue 3, Mayo 2010.

[13] Gabriel, *Cuadriculas de latitud, longitud y sistema de coordenadas*, Acolita.com, Junio 2018.

[14] Nicoletta Bloise, Elisa Capello, Matteo Dentis, Elisabetta Punta , *Obstacle Avoidance with Potential Field Appliedto a Rendezvous Maneuve*, MDPI, Applied Sciences, Octubre 2017.

[15] Jordi Cuesta Andrea, *Cinemática vectorial: velocidad angular, ángulos de Euler*, tallermatema-

tic.ovh, Agosto 2016.

[16] Autors del Text, *Precio del kWh en España en 2020*, Selectra, tarifaluzhora, Junio de 2020.

[17] Ministerios de Industria, Energía y Turismo, y Ministerio de Fomento, *FACTORES DE EMISIÓN DE CO2 y COEFICIENTES DE PASO A ENERGÍA PRIMARIA DE DIFERENTES FUENTES DE ENERGÍA FINAL CONSUMIDAS EN EL SECTOR DE EDIFICIOS EN ESPAÑA* , Julio 2014.

[18] Cristina Herrera, *¿Cuánto CO2 emites para ir al trabajo?*, eltiempo.es, Abril 2018.

Appendices

A. Apéndice I: Código del SCADA

El código principal donde se crea un objeto de la clase Cubesat es:

```
1  import controlP5.*;
2  Cubesat mycubesat;
3
4  void setup (){
5    size(1600, 1000, P3D);
6    mycubesat = new Cubesat(this);
7    mycubesat.insert();
8  }
9
10 void draw (){
11     background (255);
12     mycubesat.dataread();
13     mycubesat.orbita();
14     mycubesat.display();
15     mycubesat.botones();
16     mycubesat.Submit();
17 }
```

El código donde se programa la clase Cubesat es:

```
1  class Cubesat{
2
3  int i,colorcubesat, filas, xx, yy, zz;
4  int botonAlto=50, botonAncho=50,botonprincipalalto=50,
     botonprincipalancho=120;
5  int botonX, botonX2, botonxY=height/2-350,botonyY=height/2-200,
     botonzY=height/2-50, botonxprincipal=height/2-420,botonyprincipal=
     height/2-270,botonzprincipal=height/2-120, botonlatY=height/2+100,
```

```
    botonlatprincipal=height/2+30, botonlonY=height/2+250,
   botonlonprincipal=height/2+180;
String angx, angy, angz, bateria1, temperatura, insolacion1,
   longitudfinal1, latitudfinal1, altura1,velocidad1, epoch;
Float angX, angY , angZ, bateria, insolacion, angXdef, angYdef,
   angZdef, masa, masamenos=0.0,angxprint,angyprint, angzprint;
Table data;
int k=0,j=0;
PImage terra,satelite;
String longitudd = "0.0", latitudd = "0.0", longitudanterior1="0.0",
   masa1;
Float lon=0.0, lat=0.0, longitud = 0.0, latitud = 0.0, longitud1,
   latitud1, latitudprueba, longitudprueba, resyzex,resyzey,
   longitudfinal,latitudfinal,altura,velocidad, longitudanterior;
Table data1;
float[] latlon=new float[361], lonlat=new float[361];

ControlP5 cp5;
PImage antena;
float resyzetierrax,resyzetierray,resyzelon1,resyzelat1,resyzelon2,
   resyzelat2,resyzelon3,resyzelat3,resyzelon4,resyzelat4;
String latitudmarca,longitudmarca;
int l=1,m=1,n,botonsubmitlargo=200,botonsubmitalto=50,botonsubmitX,
   botonsubmitY1,botonsubmitY2;
String latitudtierra,longitudtierra,latituddatos1,longituddatos1,
   latituddatos2,longituddatos2,latituddatos3,longituddatos3,
   latituddatos4,longituddatos4;
float [] lattierra =new float[1000],lontierra=new float[1000],
   latdatos1=new float[10],londatos1=new float[10],latdatos2=new
   float[10],londatos2=new float[10],latdatos3=new float[10],
   londatos3=new float[10],latdatos4=new float[10],londatos4=new
   float[10];
```

```
23
24 Cubesat (PApplet thePApplet){
25     cp5 = new ControlP5(thePApplet);
26     i=0;
27     xx=0;
28     yy=0;
29     zz=0;
30     cp5.addTextfield("a").setPosition(width/2-750,height/2-400).
          setSize(100,50).setAutoClear(false);
31     cp5.addTextfield("b").setPosition(width/2-650,height/2-400).
          setSize(100,50).setAutoClear(false);
32     cp5.addTextfield("c").setPosition(width/2-750,height/2-200).
          setSize(100,50).setAutoClear(false);
33     cp5.addTextfield("d").setPosition(width/2-650,height/2-200).
          setSize(100,50).setAutoClear(false);
34     cp5.addTextfield("e").setPosition(width/2-750,height/2-100).
          setSize(100,50).setAutoClear(false);
35     cp5.addTextfield("f").setPosition(width/2-650,height/2-100).
          setSize(100,50).setAutoClear(false);
36     cp5.addTextfield("g").setPosition(width/2-750,height/2).setSize
          (100,50).setAutoClear(false);
37     cp5.addTextfield("h").setPosition(width/2-650,height/2).setSize
          (100,50).setAutoClear(false);
38     cp5.addTextfield("i").setPosition(width/2-750,height/2+100).
          setSize(100,50).setAutoClear(false);
39     cp5.addTextfield("j").setPosition(width/2-650,height/2+100).
          setSize(100,50).setAutoClear(false);
40 }
41
42  void dataread (){
43     Table data = loadTable("data.csv", "header");
44     filas= data.getRowCount();
```

```
45      angx = data.getString(i,"anguloX");
46      angy = data.getString(i,"anguloY");
47      angz = data.getString(i,"anguloZ");
48      bateria1 = data.getString(i,"bateria");
49      temperatura = data.getString(i,"temperatura");
50      insolacion1 = data.getString(i,"insolacion");
51      longitudfinal1 = data.getString(i,"longitud");
52      latitudfinal1 = data.getString(i,"latitud");
53      altura1 = data.getString(i,"altura");
54      velocidad1 = data.getString(i,"velocidad");
55      masa1=data.getString(i,"masa");
56      epoch=data.getString(i,"epoch");
57      angX = Float.parseFloat(angx);
58      angY = Float.parseFloat(angy);
59      angZ = Float.parseFloat(angz);
60      bateria = Float.parseFloat(bateria1);
61      insolacion = Float.parseFloat(insolacion1);
62      longitudfinal = Float.parseFloat(longitudfinal1);
63      latitudfinal = Float.parseFloat(latitudfinal1);
64      altura = Float.parseFloat(altura1);
65      velocidad = Float.parseFloat(velocidad1);
66      masa = Float.parseFloat(masa1);
67       if (i< filas-1){
68      i++;
69      }
70  }
71 void display() {
72    angXdef=-angX-xx;
73    angZdef=angY+yy;
74    angYdef=angZ+zz;
75    angxprint=angX+xx;
76    angyprint=angY+yy;
```

```
77  angzprint=angZ+zz;
78  fill(255,100,0);
79  rect(width/2-220, height/2-425, 220, 390,10);
80  rect(width/2-210,height/2+390, 730, 40, 10);
81  fill(255,255,0);
82  rect(width/2-210, height/2-415, 200, 20,10);
83  rect(width/2-210, height/2-365, 200, 20,10);
84  rect(width/2-210, height/2-315, 200, 20,10);
85  rect(width/2-210, height/2-265, 200, 20,10);
86  rect(width/2-210, height/2-215, 200, 20,10);
87  rect(width/2-210, height/2-165, 200, 20,10);
88  rect(width/2-210, height/2-115, 200, 20,10);
89  rect(width/2-210, height/2-65, 200, 20,10);
90  rect(width/2-200, height/2+400, 170, 20, 10);
91  rect(width/2-20, height/2+400, 170, 20, 10);
92  rect(width/2+160, height/2+400, 170, 20, 10);
93  rect(width/2+340, height/2+400, 170, 20, 10);
94  fill(0);
95  text("Orientaci n␣Cubesat", width/2+200, height/2-470);
96  text("Datos␣Cubesat", width/2-180, height/2-450);
97  text("Control␣Cubesat", botonX+20, botonxprincipal-20);
98  text("Traza␣terrestre", width/2+100, height/2-10);
99  text("ANGULO␣X:␣"+angxprint+"  ", width/2-200, height/2-350);
100 text("ANGULO␣Y:␣"+angyprint+"  ", width/2-200, height/2-300);
101 text("ANGULO␣Z:␣"+angzprint+"  ", width/2-200, height/2-250);
102 text("BATERIA:␣"+bateria+"%", width/2-200, height/2-200);
103 text("TEMPERATURA:␣"+temperatura+"  ", width/2-200, height/2-150);
104 if(insolacion==1){
105   text("CUBESAT␣INSOLADO", width/2-200, height/2-100);
106 }
107 else{
108 text("CUBESAT␣NO␣INSOLADO", width/2-200, height/2-100);
```

```
109   }
110   text("MASA:␣"+masa+"kg", width/2-200, height/2-50);
111   text("FECHA:␣"+epoch+"", width/2-200, height/2-400);
112   text("LONGITUD:␣"+longitud1+"  ",width/2-190, height/2+415);
113   text("LATITUD:␣"+latitud1+"  ",width/2-10, height/2+415);
114   text("ALTURA:␣"+altura/1000+"km",width/2+170, height/2+415);
115   text("VELOCIDAD:␣"+velocidad/1000+"km/s",width/2+350, height/2+415)
        ;
116   if (bateria<=5){
117   colorcubesat=0;
118   }
119   else{
120   colorcubesat=255;
121   }
122   //ejes
123   stroke(0);
124   line(width/2+50, height/2-250,0, width/2+450,height/2-250,0);
125   line(width/2+250, height/2-50,0, width/2+250,height/2-450,0);
126   line(width/2+250, height/2-250,-1500, width/2+250,height/2-250,350)
        ;
127   text("X", width/2+450,height/2-245);
128   text("Y", width/2+247,height/2-452+420);
129   text("Z", width/2+83, height/2-85);
130   // Creaci n del rectangulo 3D
131   pushMatrix();
132   stroke(0);
133   lights();
134   translate(width/2+250, height/2-250, 0);
135   scale(5, 5, 5);
136   rotateX(radians(angXdef));
137   rotateZ(radians(angYdef));
138   rotateY(radians(angZdef));
```

```
139   beginShape(QUADS);
140   //Cara delantera
141   fill(255, colorcubesat, 0);
142   vertex(-20, -20, 20);
143   vertex(20, -20, 20);
144   vertex(20, 20, 20);
145   vertex(-20, 20, 20);
146   //Cara trasera
147   fill(255, colorcubesat, 0);
148   vertex(-20, -20, -20);
149   vertex(20, -20, -20);
150   vertex(20, 20, -20);
151   vertex(-20, 20, -20);
152   //Cara lateral izquierda
153   fill(255, colorcubesat, 0);
154   vertex(-20, -20, -20);
155   vertex(-20, -20, 20);
156   vertex(-20, 20, 20);
157   vertex(-20, 20, -20);
158   //Cara lateral derecha
159   fill(255, colorcubesat, 0);
160   vertex(20, -20, -20);
161   vertex(20, -20, 20);
162   vertex(20, 20, 20);
163   vertex(20, 20, -20);
164   //cara superior
165   fill(255, colorcubesat, 0);
166   vertex(-20, -20, -20);
167   vertex(20, -20, -20);
168   vertex(20, -20, 20);
169   vertex(-20, -20, 20);
170   //Cara inferior
```

```
171   fill(255, colorcubesat, 0);
172   vertex(-20, 20, -20);
173   vertex(20, 20, -20);
174   vertex(20, 20, 20);
175   vertex(-20, 20, 20);
176   endShape();
177   beginShape(LINES);
178   vertex(-40, 0,0);
179   vertex(40,0,0);
180   stroke(100);
181   vertex(0,-40,0);
182   vertex(0,40,0);
183   stroke(200);
184   vertex(0, 0,-40);
185   vertex(0,0,40);
186   endShape();
187   popMatrix();
188 }
189
190 void botones (){
191
192   botonX=width/2-450;
193   botonX2=width/2-380;
194   botonxY=height/2-350;
195   botonyY=height/2-200;
196   botonzY=height/2-50;
197   botonxprincipal=height/2-420;
198   botonyprincipal=height/2-270;
199   botonzprincipal=height/2-120;
200   botonlatY=height/2+100;
201   botonlatprincipal=height/2+30;
202   botonlonY=height/2+250;
```

```
203  botonlonprincipal=height/2+180;
204
205  fill(255,100,0);
206  rect(width/2-460, height/2-430, 140, 740,10);
207
208  fill(20);
209  rect(botonX,botonxprincipal,botonprincipalancho,botonprincipalalto
        ,10);
210  fill(255);
211  text(" ngulo ␣X", botonX+35, botonxprincipal+25);
212
213  fill(20);
214  rect(botonX,botonxY,botonAncho,botonAlto,10);
215  fill(255);
216  text("+", botonX+22, botonxY+28);
217  if((mouseX<botonX+botonAncho)&&(mouseX>botonX)&&(mouseY<botonxY+
        botonAlto)&&(mouseY>botonxY&&mousePressed)){
218  xx++;
219   masamenos=masamenos-0.005;
220  }
221
222  fill(20);
223  rect(botonX2,botonxY,botonAncho,botonAlto,10);
224  fill(255);
225  text("-", botonX2+22, botonxY+28);
226  if(mouseX<botonX2+botonAncho&&mouseX>botonX2&&mouseY<botonxY+
        botonAlto&&mouseY>botonxY&&mousePressed){
227  xx--;
228    masamenos=masamenos-0.005;
229  }
230
231  fill(20);
```

```
232  rect(botonX,botonyprincipal,botonprincipalancho,botonprincipalalto
        ,10);
233  fill(255);
234  text(" ngulo ␣Y", botonX+35, botonyprincipal+25);
235
236  fill(20);
237  rect(botonX,botonyY,botonAncho,botonAlto,10);
238  fill(255);
239  text("+", botonX+22, botonyY+28);
240  if(mouseX<botonX+botonAncho&&mouseX>botonX&&mouseY<botonyY+
        botonAlto&&mouseY>botonyY&&mousePressed){
241  yy++;
242    masamenos=masamenos-0.005;
243  }
244
245  fill(20);
246  rect(botonX2,botonyY,botonAncho,botonAlto,10);
247  fill(255);
248  text("-", botonX2+22, botonyY+28);
249  if(mouseX<botonX2+botonAncho&&mouseX>botonX2&&mouseY<botonyY+
        botonAlto&&mouseY>botonyY&&mousePressed){
250  yy--;
251    masamenos=masamenos-0.005;
252  }
253
254  fill(20);
255  rect(botonX,botonzprincipal,botonprincipalancho,botonprincipalalto
        ,10);
256  fill(255);
257  text(" ngulo ␣Z", botonX+35, botonzprincipal+25);
258
259  fill(20);
```

```
260  rect(botonX,botonzY,botonAncho,botonAlto,10);
261  fill(255);
262  text("+", botonX+22, botonzY+28);
263  if(mouseX<botonX+botonAncho&&mouseX>botonX&&mouseY<botonzY+
        botonAlto&&mouseY>botonzY&&mousePressed){
264  zz++;
265   masamenos=masamenos-0.005;
266  }
267
268  fill(20);
269  rect(botonX2,botonzY,botonAncho,botonAlto,10);
270  fill(255);
271  text("-", botonX2+22, botonzY+28);
272  if(mouseX<botonX2+botonAncho&&mouseX>botonX2&&mouseY<botonzY+
        botonAlto&&mouseY>botonzY&&mousePressed){
273  zz--;
274   masamenos=masamenos-0.005;
275  }
276
277   fill(20);
278  rect(botonX,botonlatprincipal,botonprincipalancho,
        botonprincipalalto,10);
279  fill(255);
280  text("LATITUD", botonX+35, botonlatprincipal+25);
281
282  fill(20);
283  rect(botonX,botonlatY,botonAncho,botonAlto,10);
284  fill(255);
285  text("+", botonX+22, botonlatY+28);
286  if(mouseX<botonX+botonAncho&&mouseX>botonX&&mouseY<botonlatY+
        botonAlto&&mouseY>botonlatY&&mousePressed){
287  lat++;
```

```
288  masamenos=masamenos-0.005;
289  }
290
291  fill(20);
292  rect(botonX2,botonlatY,botonAncho,botonAlto,10);
293  fill(255);
294  text("-", botonX2+22, botonlatY+28);
295  if(mouseX<botonX2+botonAncho&&mouseX>botonX2&&mouseY<botonlatY+
        botonAlto&&mouseY>botonlatY&&mousePressed){
296  lat--;
297  masamenos=masamenos-0.005;
298  }
299
300     fill(20);
301  rect(botonX,botonlonprincipal,botonprincipalancho,
        botonprincipalalto,10);
302  fill(255);
303  text("LONGITUD", botonX+35, botonlonprincipal+25);
304
305  fill(20);
306  rect(botonX,botonlonY,botonAncho,botonAlto,10);
307  fill(255);
308  text("+", botonX+22, botonlonY+28);
309  if(mouseX<botonX+botonAncho&&mouseX>botonX&&mouseY<botonlonY+
        botonAlto&&mouseY>botonlonY&&mousePressed){
310  lon++;
311  masamenos=masamenos-0.005;
312  }
313
314  fill(20);
315  rect(botonX2,botonlonY,botonAncho,botonAlto,10);
316  fill(255);
```

```
317   text("-", botonX2+22, botonlonY+28);
318   if(mouseX<botonX2+botonAncho&&mouseX>botonX2&&mouseY<botonlonY+
          botonAlto&&mouseY>botonlonY&&mousePressed){
319   lon--;
320   masamenos=masamenos-0.005;
321   }
322 }
323
324 void orbita() {
325     latlon[i-1]=lat;
326     lonlat[i-1]=lon;
327     if(i==filas-1){
328     latlon[i]=lat;
329     lonlat[i]=lon;
330     }
331     terra=loadImage("mapa.jpg");
332     satelite=loadImage("satelitee.png");
333     terra.resize(750,376);
334     satelite.resize(20,20);
335   image(terra,width/2-220,height/2);
336   Table data1 = loadTable("data.csv", "header");
337   for(j=0;j<=k;j++){
338       // Accedemos a cada uno de los par metros enviados.
339       longitudd = data1.getString(j,"longitud");
340       latitudd = data1.getString(j,"latitud");
341
342       // Convertimos los datos recibidos en valores num ricos.
343       longitud = Float.parseFloat(longitudd);
344       latitud = Float.parseFloat(latitudd);
345       latitudprueba=latitud+latlon[j];
346       longitudprueba=longitud+lonlat[j];
347
```

```
348       if(longitudprueba<-180){
349       longitudanterior1 = data1.getString(j-1,"longitud");
350       longitudanterior = Float.parseFloat(longitudanterior1);
351       if(longitudanterior>0&&longitud<0){
352       lon=longitudanterior+lonlat[j-1]-longitud;
353       lonlat[j]=lon;
354       }
355         else{
356       lon=180-longitud;
357       lonlat[j]=lon;
358         }
359       }
360       else if (longitudprueba>180){
361       lon=-180-longitud;
362       lonlat[j]=lon;
363       }
364
365        if(latitudprueba<-90){
366       lat=90-latitud;
367       latlon[j]=lat;
368       }
369       else if (latitudprueba>90){
370       lat=-90-latitud;
371       latlon[j]=lat;
372       }
373
374       latitud1=latitud+latlon[j];
375       longitud1=longitud+lonlat[j];
376
377   resyzex=longitud1/180*375+width/2+155;
378   resyzey=-latitud1/90*188+height/2+188;
379   noStroke();
```

```
380   fill(255,0,0);
381   rect(resyzex,resyzey,4,4);
382   }
383   if (k< filas-1){
384     k++;
385     }
386   image(satelite,resyzex,resyzey-8);
387 }
388
389 void insert (){
390 }
391
392 void Submit(){
393   fill(20);
394   text("INTRODUCIR␣ESTACIONES␣TERRESTRES",35, 60);
395   text("INTRODUCIR␣ REA ␣DE␣ESTUDIO",50, 260);
396   text("LONGITUD",50, 90);
397   text("LATITUD", 150, 90);
398   text("LONGITUD␣1",50, 290);
399   text("LATITUD␣1", 150, 290);
400   text("LONGITUD␣2",50, 390);
401   text("LATITUD␣2", 150, 390);
402   text("LONGITUD␣3",50, 490);
403   text("LATITUD␣3", 150, 490);
404   text("LONGITUD␣4",50, 590);
405   text("LATITUD␣4", 150, 590);
406   botonsubmitX=50;
407   botonsubmitY1=175;
408   botonsubmitY2=675;
409   fill(20);
410   rect(botonsubmitX,botonsubmitY1,botonsubmitlargo,botonsubmitalto
        ,10);
```

```
411   rect(botonsubmitX,botonsubmitY2,botonsubmitlargo,botonsubmitalto
          ,10);
412   fill(255);
413   text("ENVIAR", botonsubmitX+80, botonsubmitY1+28);
414   text("ENVIAR", botonsubmitX+80, botonsubmitY2+28);
415   if(mouseX<botonsubmitX+botonsubmitlargo&&mouseX>botonsubmitX&&
          mouseY<botonsubmitY1+botonsubmitalto&&mouseY>botonsubmitY1&&
          mousePressed){
416     latitudmarca=cp5.get(Textfield.class,"b").getText();
417     longitudmarca=cp5.get(Textfield.class,"a").getText();
418     lattierra[l]=Float.parseFloat(latitudmarca);
419     lontierra[l]=Float.parseFloat(longitudmarca);
420     l++;
421     }
422       if(mouseX<botonsubmitX+botonsubmitlargo&&mouseX>botonsubmitX&&
             mouseY<botonsubmitY2+botonsubmitalto&&mouseY>botonsubmitY2&&
             mousePressed){
423     latituddatos1=cp5.get(Textfield.class,"d").getText();
424     longituddatos1=cp5.get(Textfield.class,"c").getText();
425     latituddatos2=cp5.get(Textfield.class,"f").getText();
426     longituddatos2=cp5.get(Textfield.class,"e").getText();
427     latituddatos3=cp5.get(Textfield.class,"h").getText();
428     longituddatos3=cp5.get(Textfield.class,"g").getText();
429     latituddatos4=cp5.get(Textfield.class,"j").getText();
430     longituddatos4=cp5.get(Textfield.class,"i").getText();
431     latdatos1[m]=Float.parseFloat(latituddatos1);
432     londatos1[m]=Float.parseFloat(longituddatos1);
433     latdatos2[m]=Float.parseFloat(latituddatos2);
434     londatos2[m]=Float.parseFloat(longituddatos2);
435     latdatos3[m]=Float.parseFloat(latituddatos3);
436     londatos3[m]=Float.parseFloat(longituddatos3);
437     latdatos4[m]=Float.parseFloat(latituddatos4);
```

```
438      londatos4[m]=Float.parseFloat(longituddatos4);
439      m++;
440      }
441    for (n=1;n<=l-1;n++){
442        resyzetierrax=lontierra[n]/180*375+width/2+155;
443        resyzetierray=-lattierra[n]/90*188+height/2+188;
444        antena=loadImage("antena.png");
445        antena.resize(20,20);
446        image(antena,resyzetierrax-10,resyzetierray-10);
447    }
448     for (n=1;n<=m-1;n++){
449        resyzelon1=londatos1[n]/180*375+width/2+155;
450        resyzelat1=-latdatos1[n]/90*188+height/2+188;
451        resyzelon2=londatos2[n]/180*375+width/2+155;
452        resyzelat2=-latdatos2[n]/90*188+height/2+188;
453        resyzelon3=londatos3[n]/180*375+width/2+155;
454        resyzelat3=-latdatos3[n]/90*188+height/2+188;
455        resyzelon4=londatos4[n]/180*375+width/2+155;
456        resyzelat4=-latdatos4[n]/90*188+height/2+188;
457        line(resyzelon1,resyzelat1,resyzelon2,resyzelat2);
458        line(resyzelon2,resyzelat2,resyzelon3,resyzelat3);
459        line(resyzelon3,resyzelat3,resyzelon4,resyzelat4);
460        line(resyzelon4,resyzelat4,resyzelon1,resyzelat1);
461        }
462 }
463
464 }
```